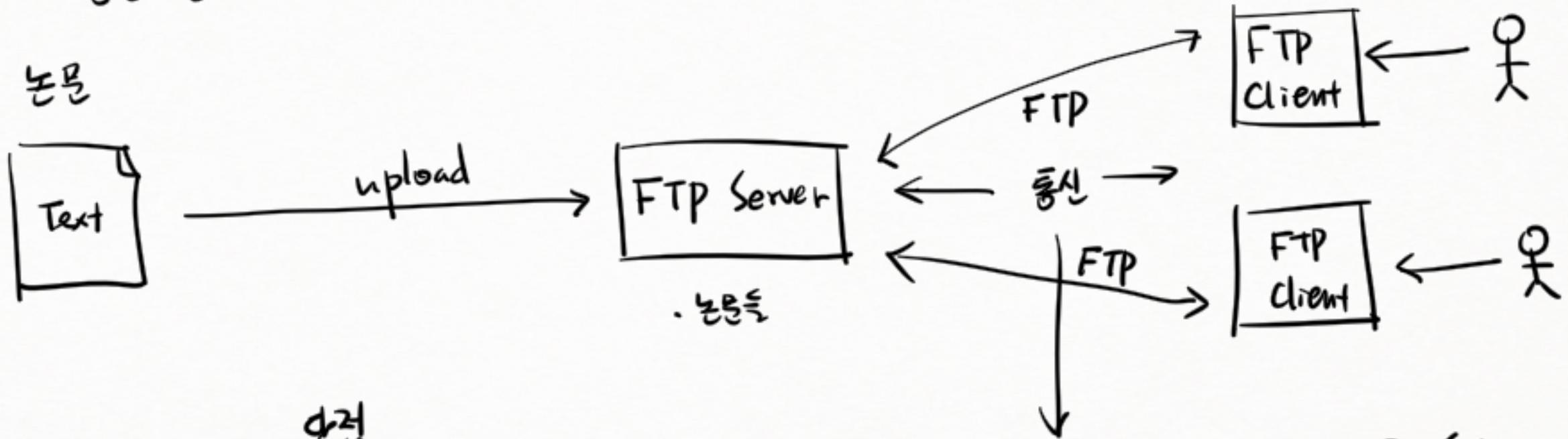


* HTML, HTTP

① HTTP, HTML 등장 전



단점

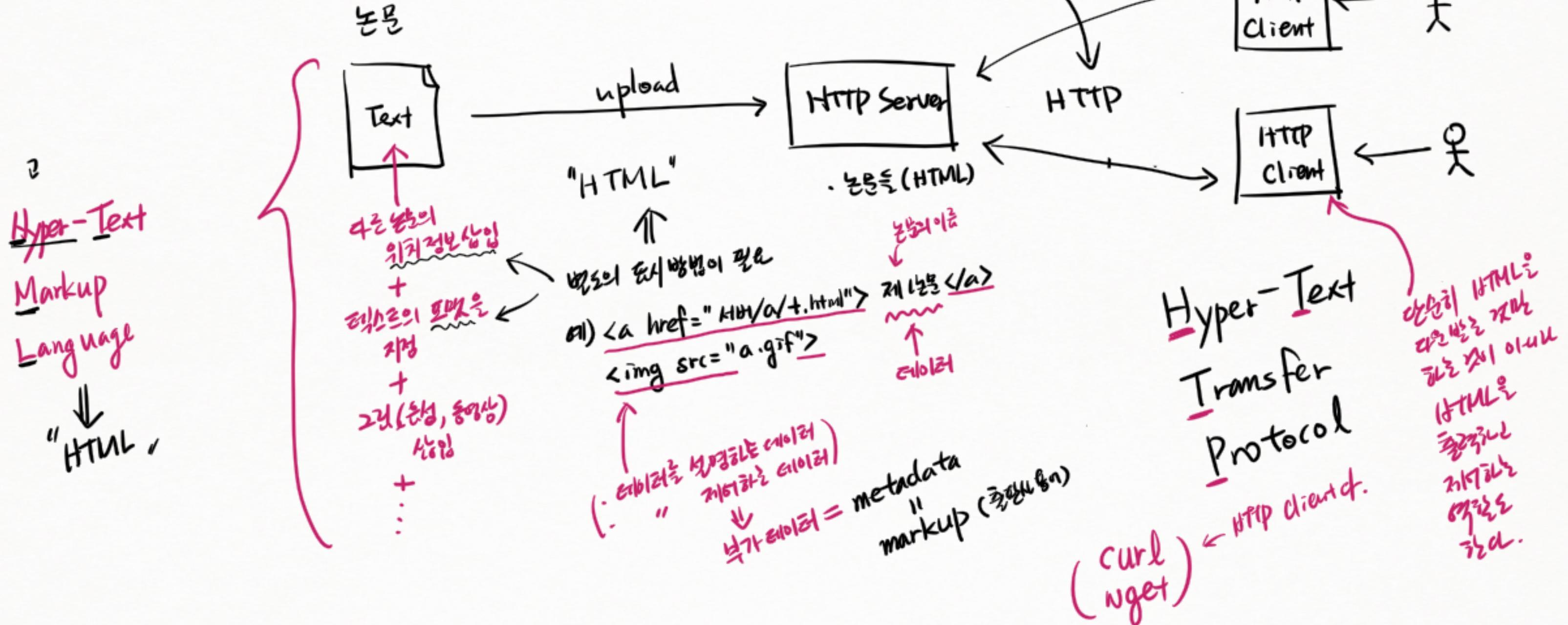
- 논문이 첨부하는 다른 논문을
다운로드하기 번거롭다
- 논문 안에 다른 논문이 업로드된
서버 주소가 많다.

통신 규칙이 아니라 Data를 송수신
"protocol"

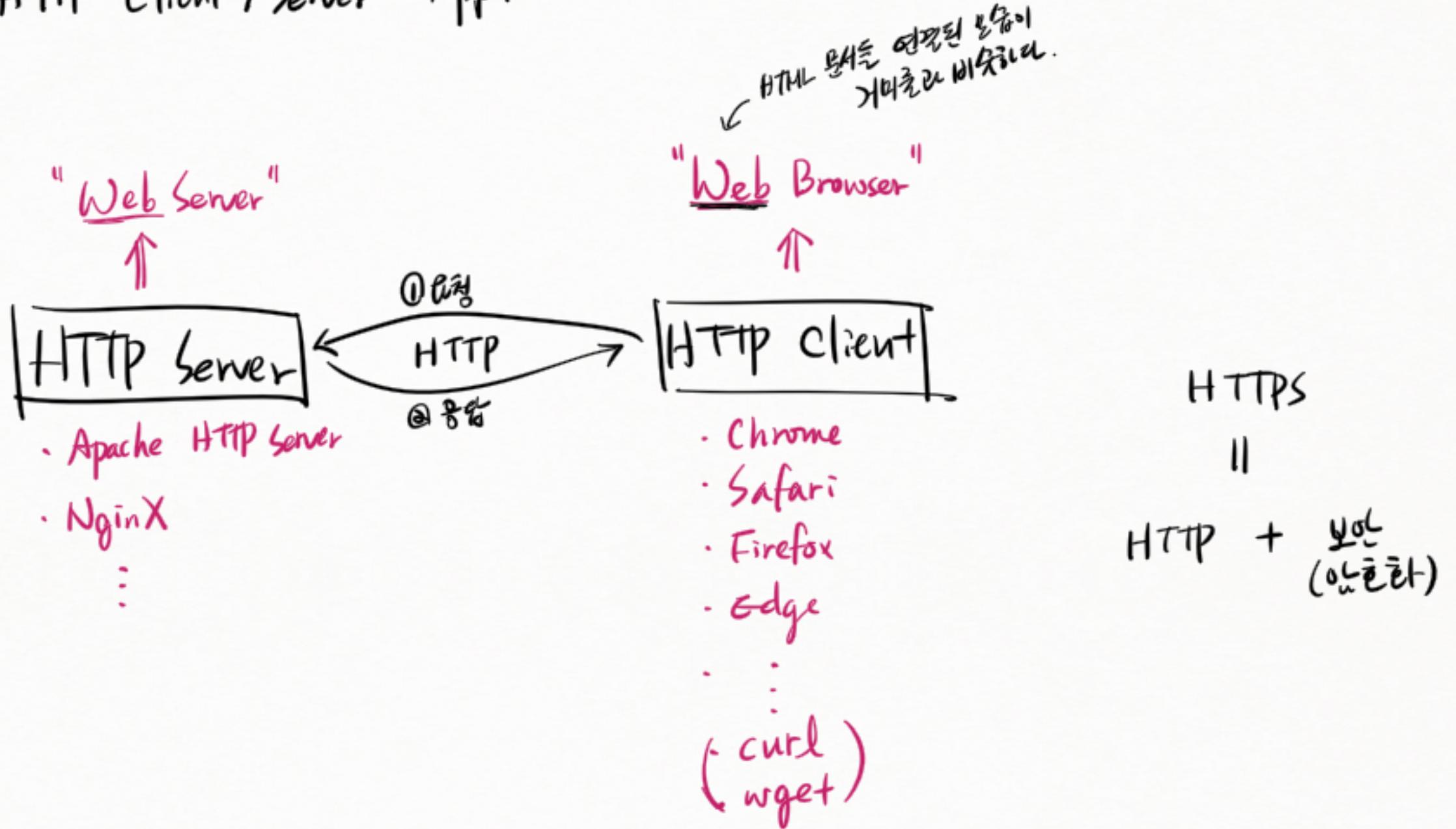
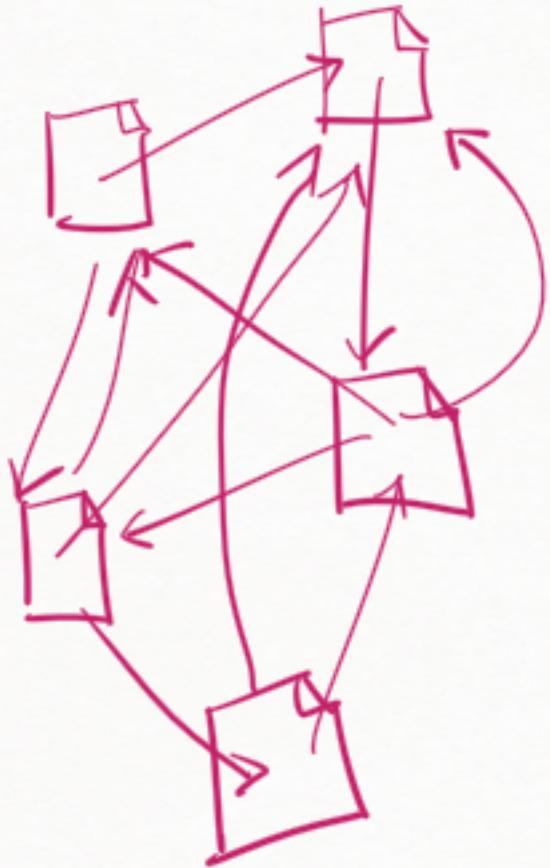
File Transfer Protocol

* HTML, HTTP

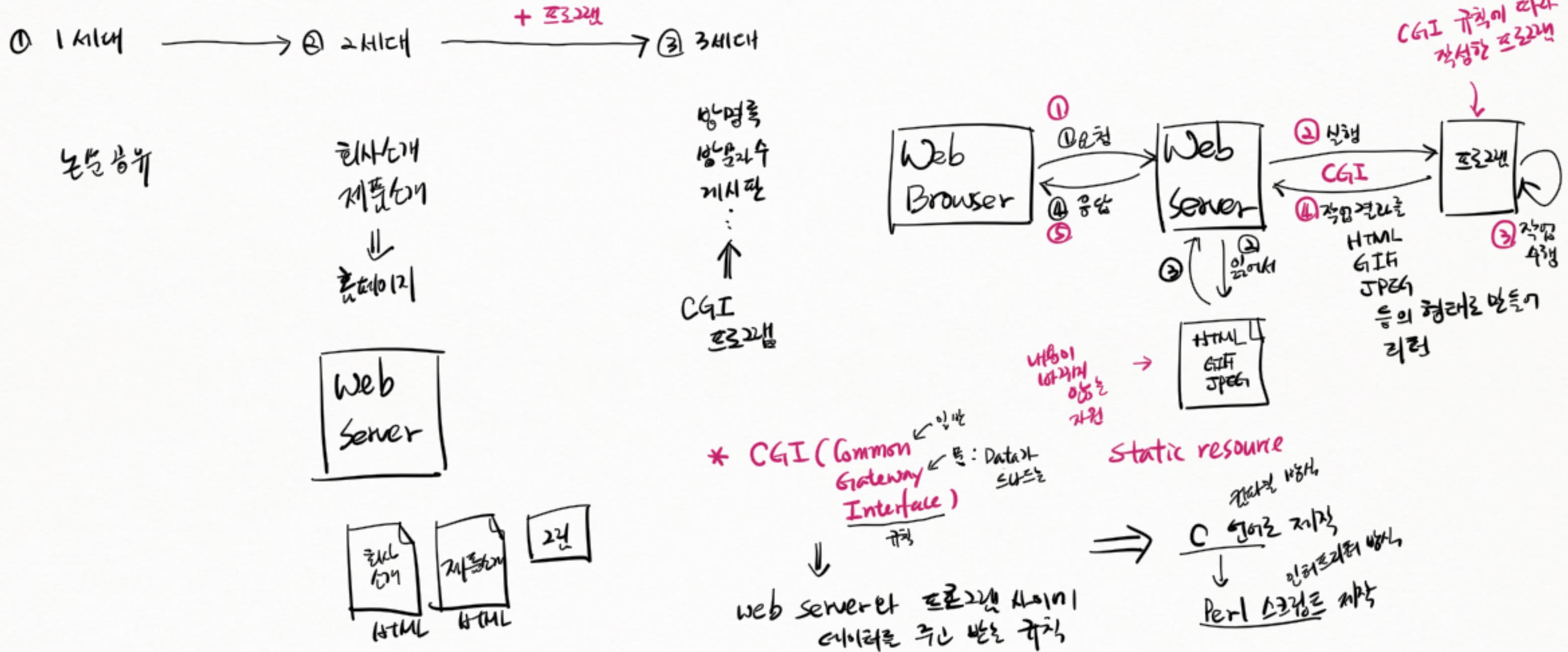
② HTTP, HTML 등장



* HTTP Client / Server App.



* Web 기술의 활용

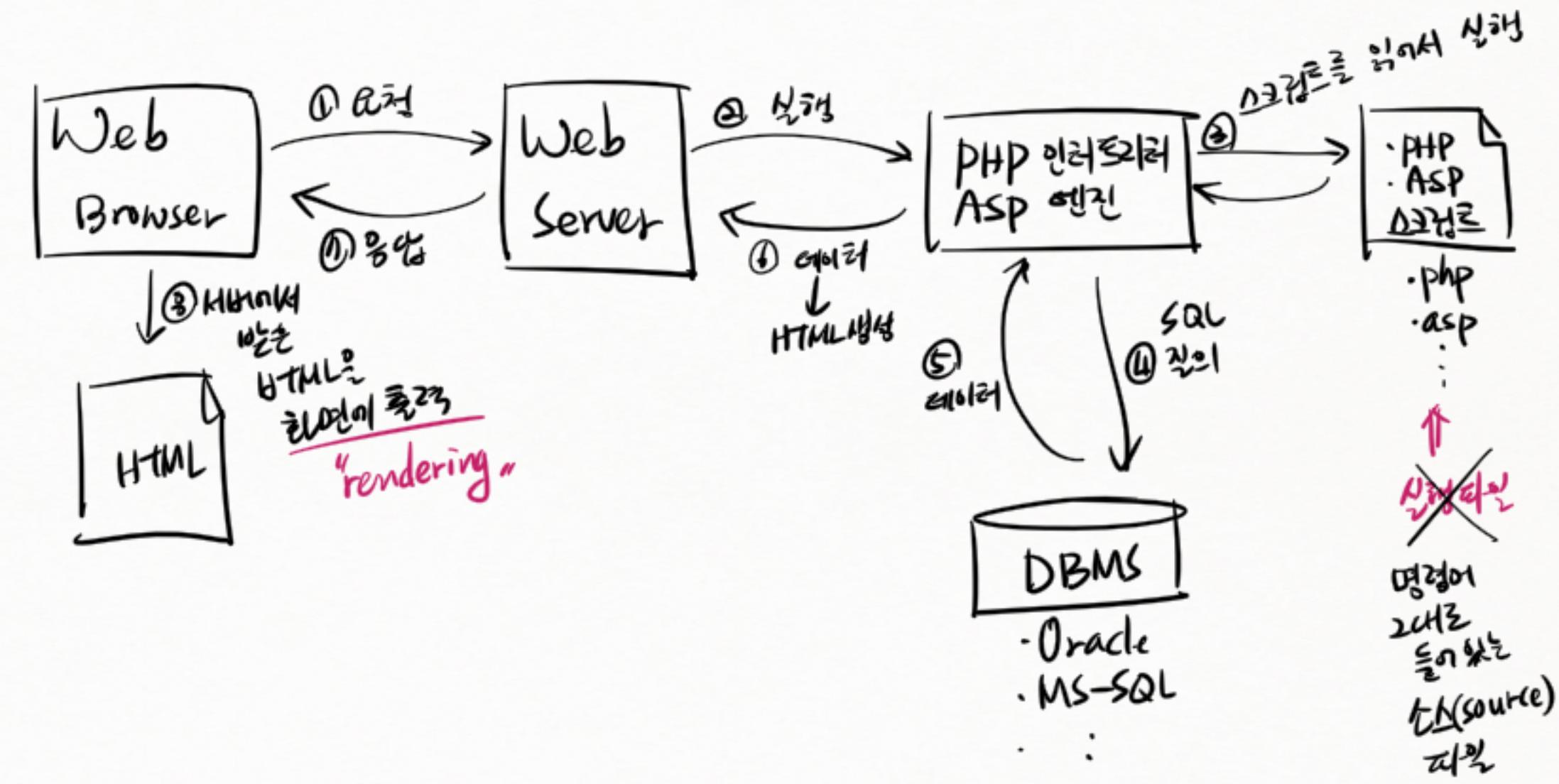


* Web 기술의 활용 II

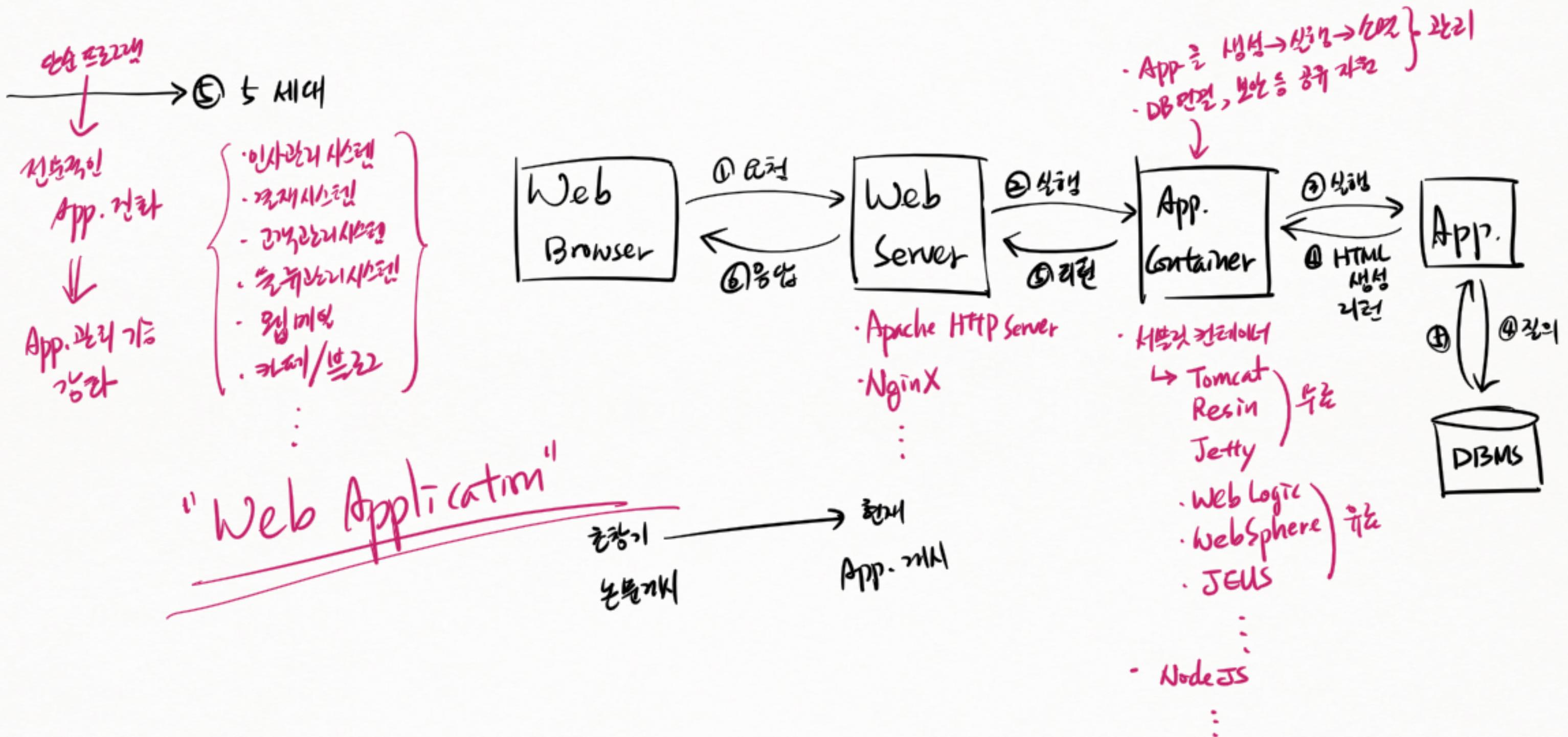
+ DBMS
+ 스크립트
→ ④ 웹서버

수정물
+
MS ASP 프로그래밍
PHP
:
+
DBMS

(01) 웹프로그래밍



* Web 기술의 활용 III



* Web Application 2가지 기술

[전면 만드는 기술]

CSS — 웹컨트의 디자인을
제작하는 기술

+

HTML ← 웹컨트를 구성하는
내용하는 기술

+

JavaScript ←
• 사용자의 입력에 응답
• 웹컨트를 제어

:

“Front-end 2가지”

← Full Stack
2가지

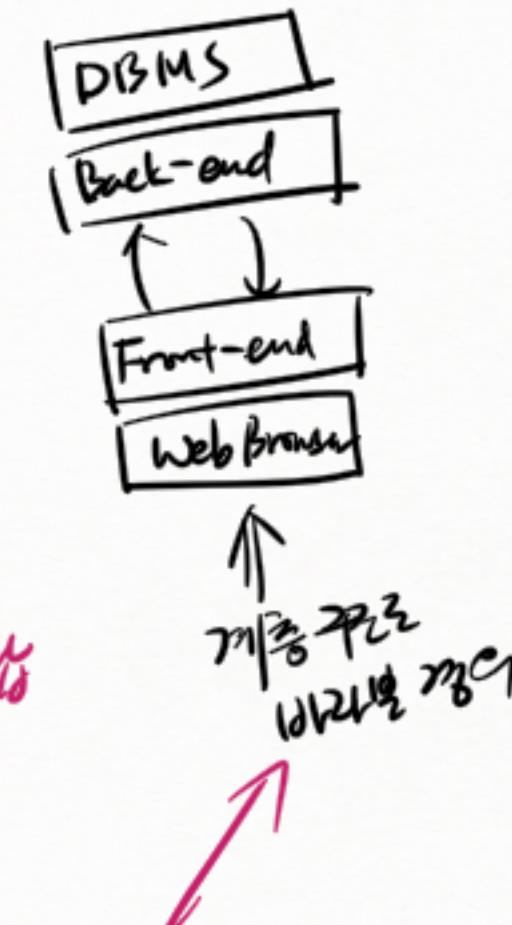
SI / SM

[서버를 만드는 기술]

✓ Java, JavaScript, PHP, ...

✓ SQL

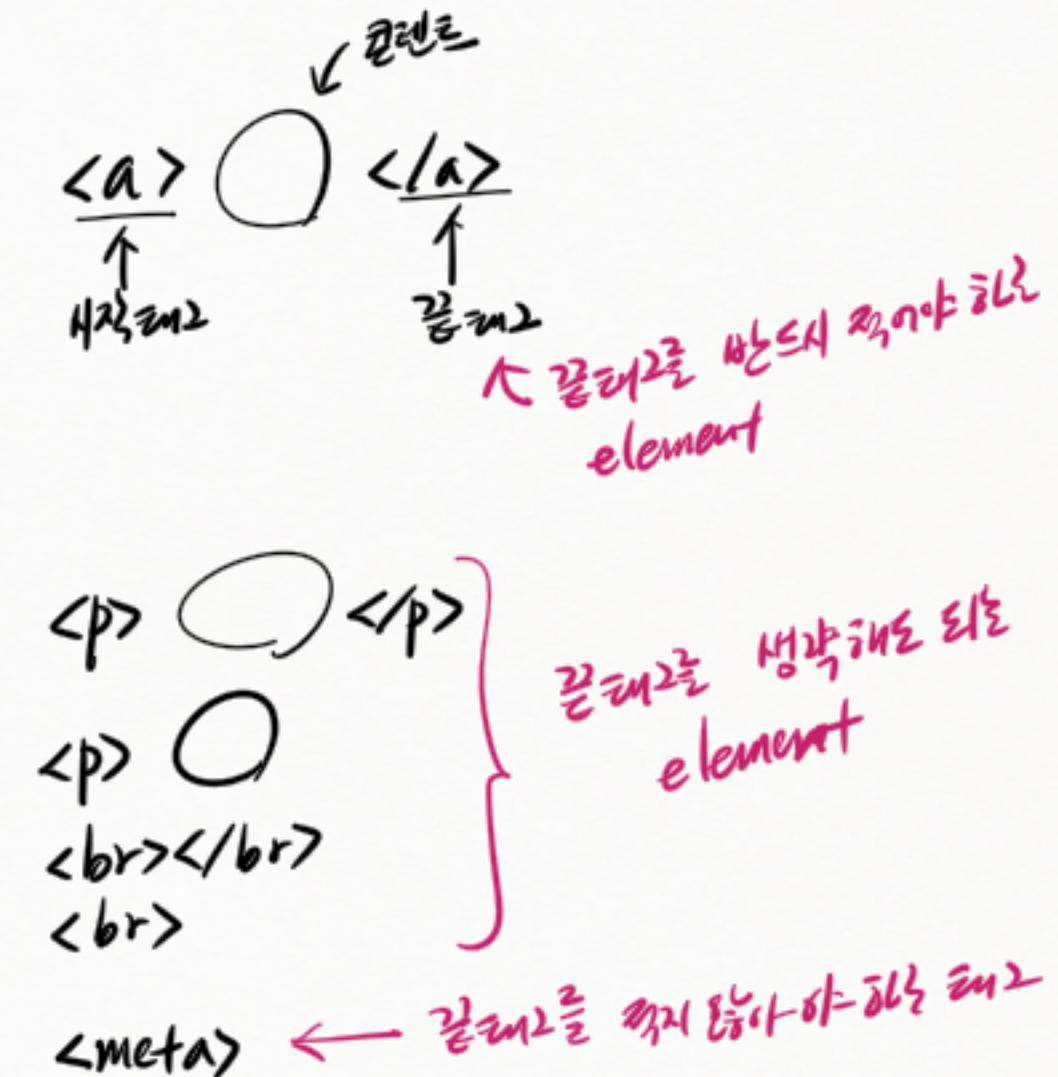
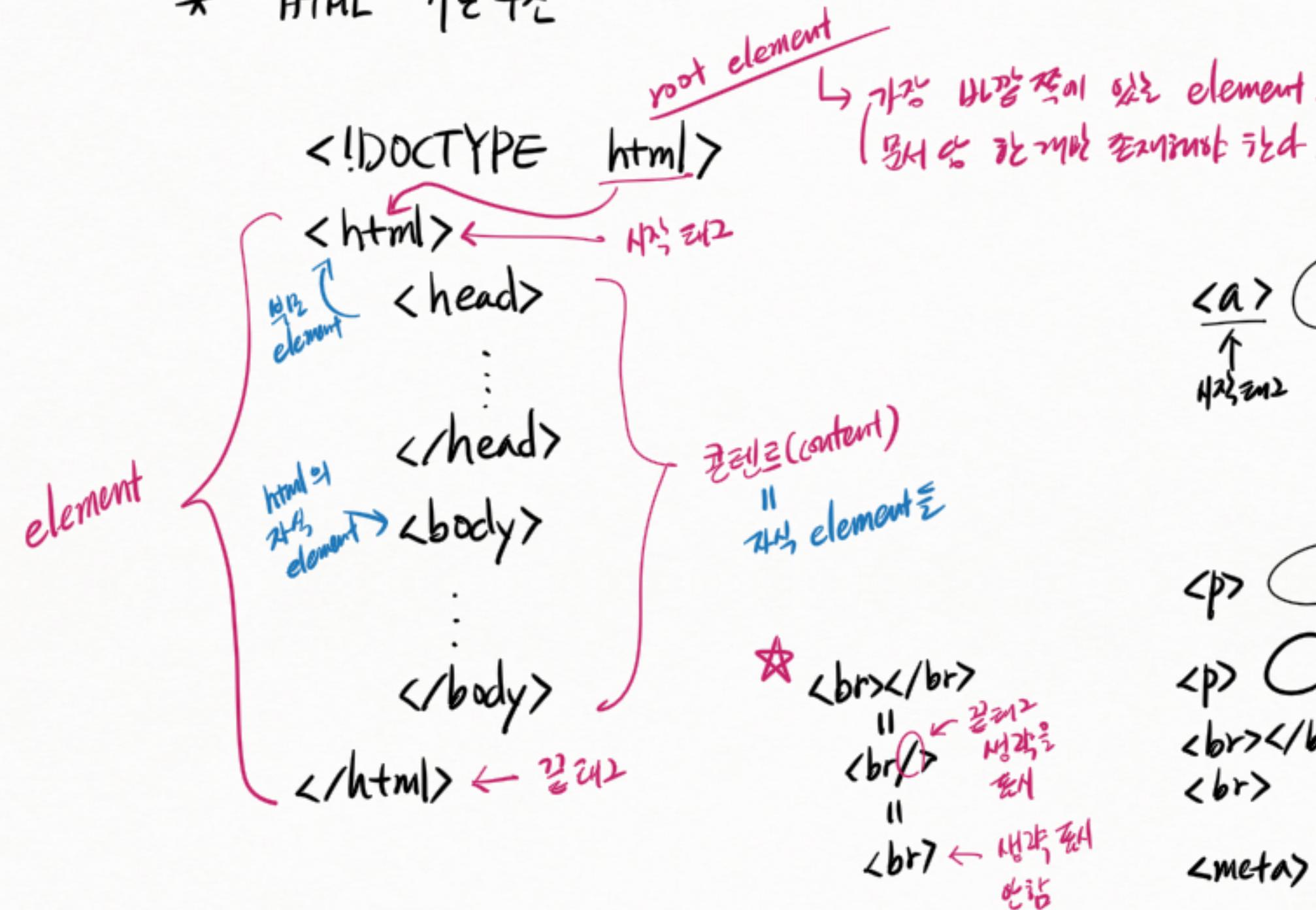
:



“Back-end 2가지”

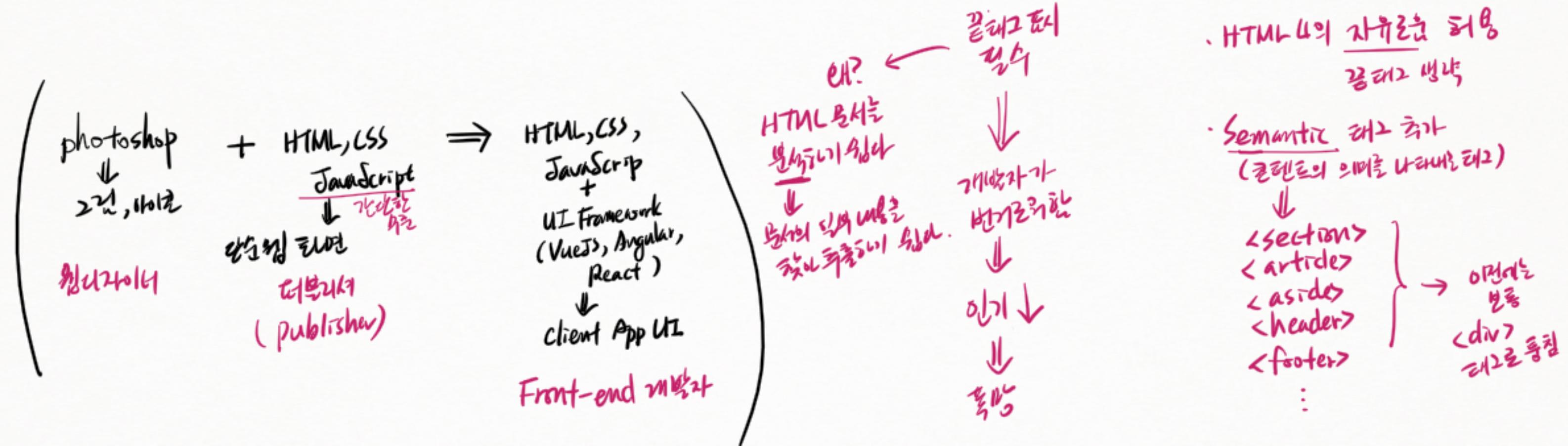
HTML

* HTML 기본 구조

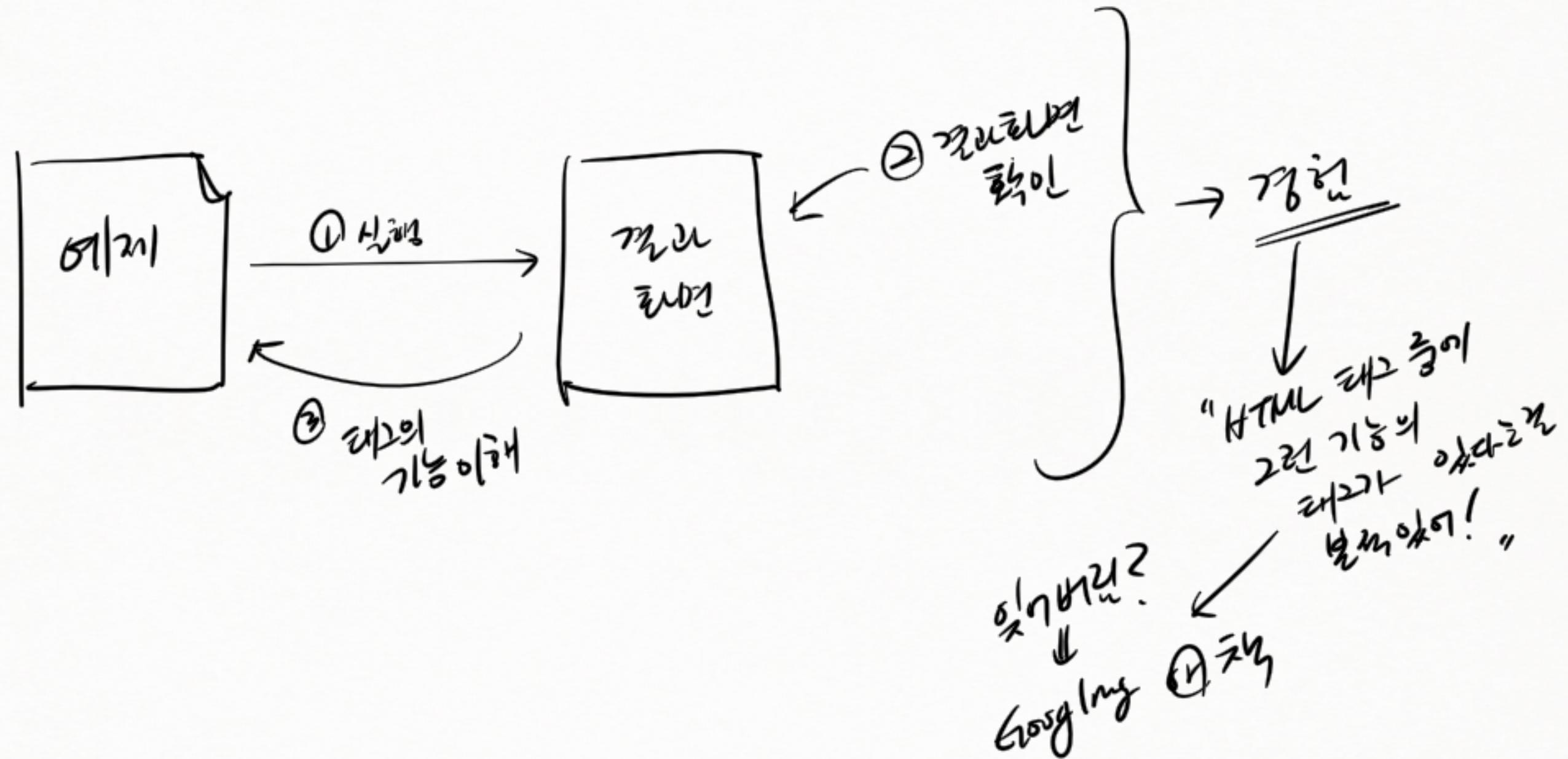


* HTML

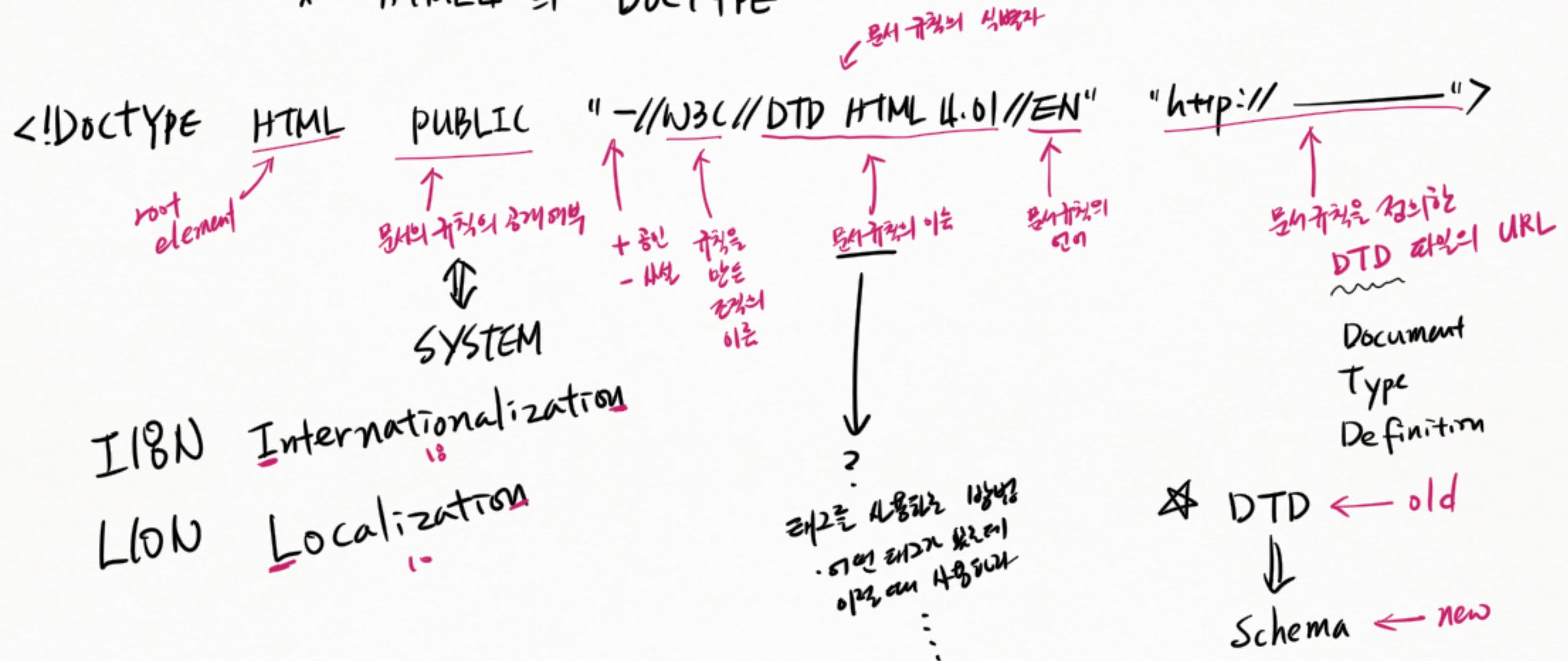
HTML 3.x → HTML 4.x → XHTML → HTML5



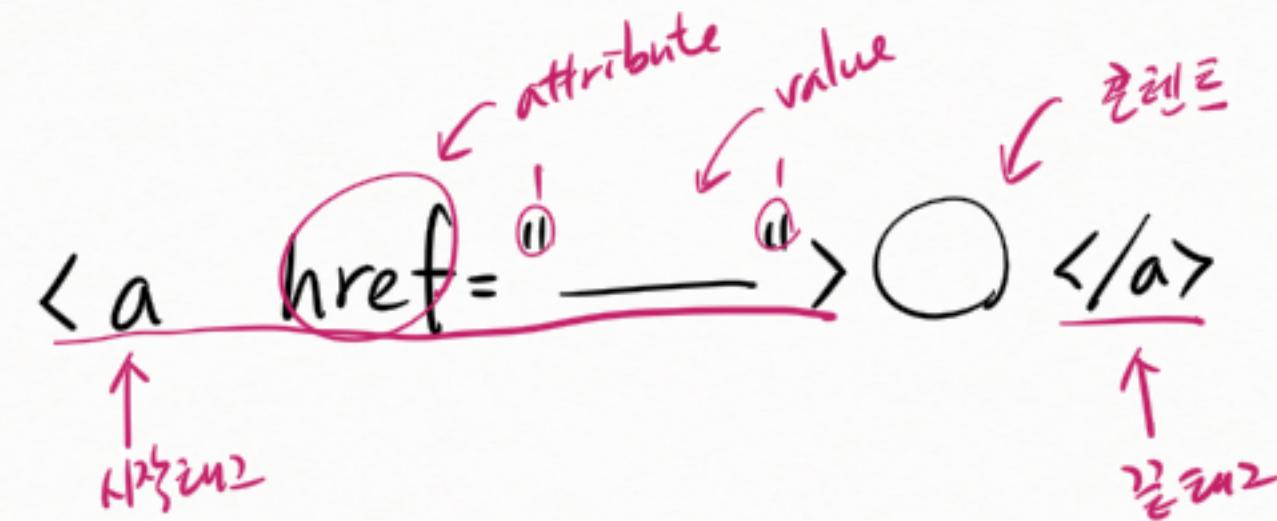
* HTML 풍부함



* HTML4 의 DOCTYPE



* tag et attribute



controls
selected
checked
readonly
:
} 키보드에
 마우스에
 상호작용이
 제어권으로
 기능을
 제공하는
 특수한 속성

* CSS

① <style> 태그

```
<style>
  =
</style>
```

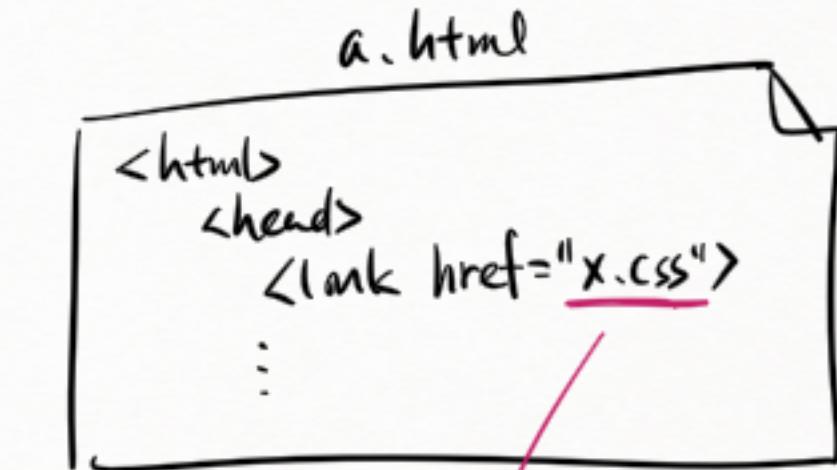
② inline 디자인

```
<tfoot style="—————" > 0 </tfoot>
```

inline style

이것!

③ 외부 css 적용



외부 HTML 파일에서
css를どのように 사용하는가?

x.css



* MIME Type

Multi-purpose
Internet
Mail
Extensions

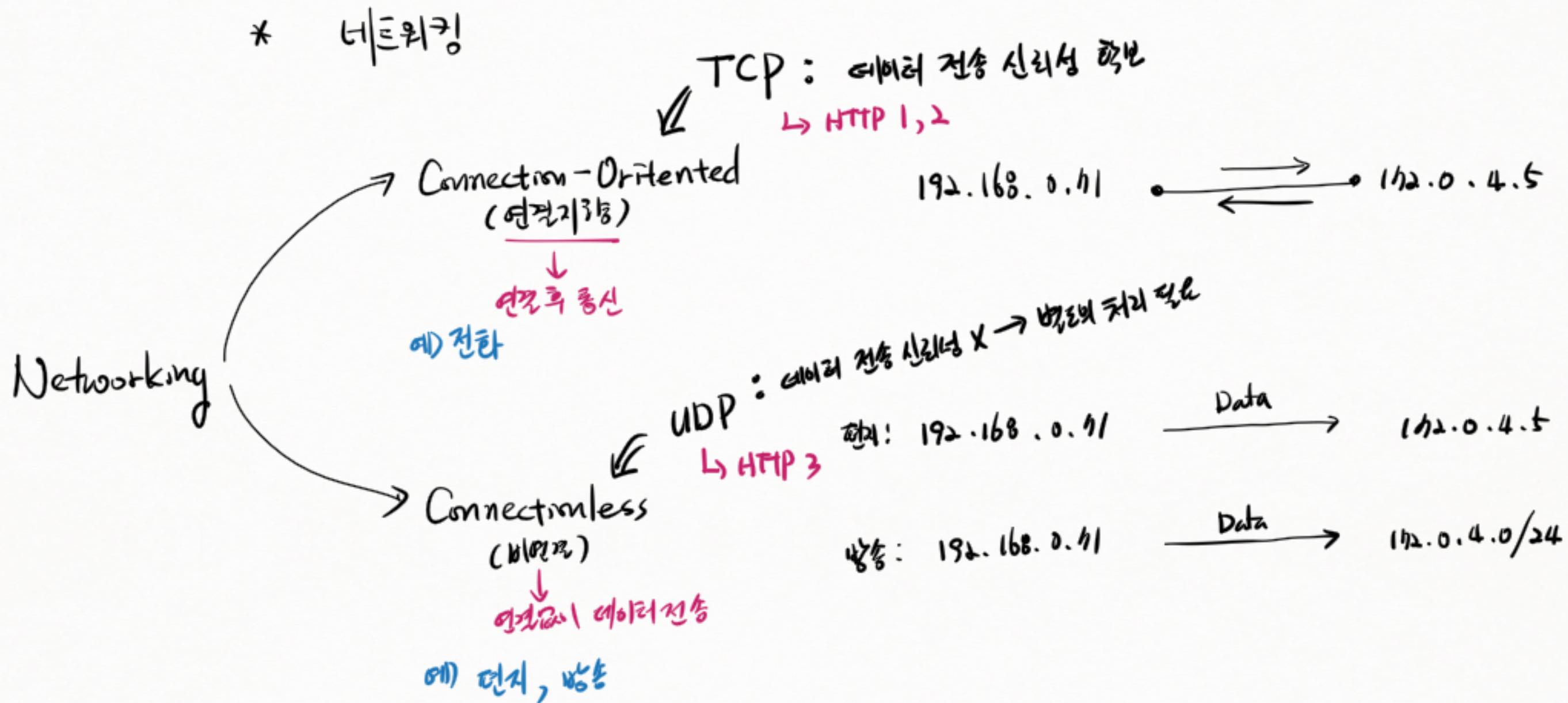
}
메일이 전부는 초 텐트가 어떤 형식인지
인터넷이나 메일과 같은 형식



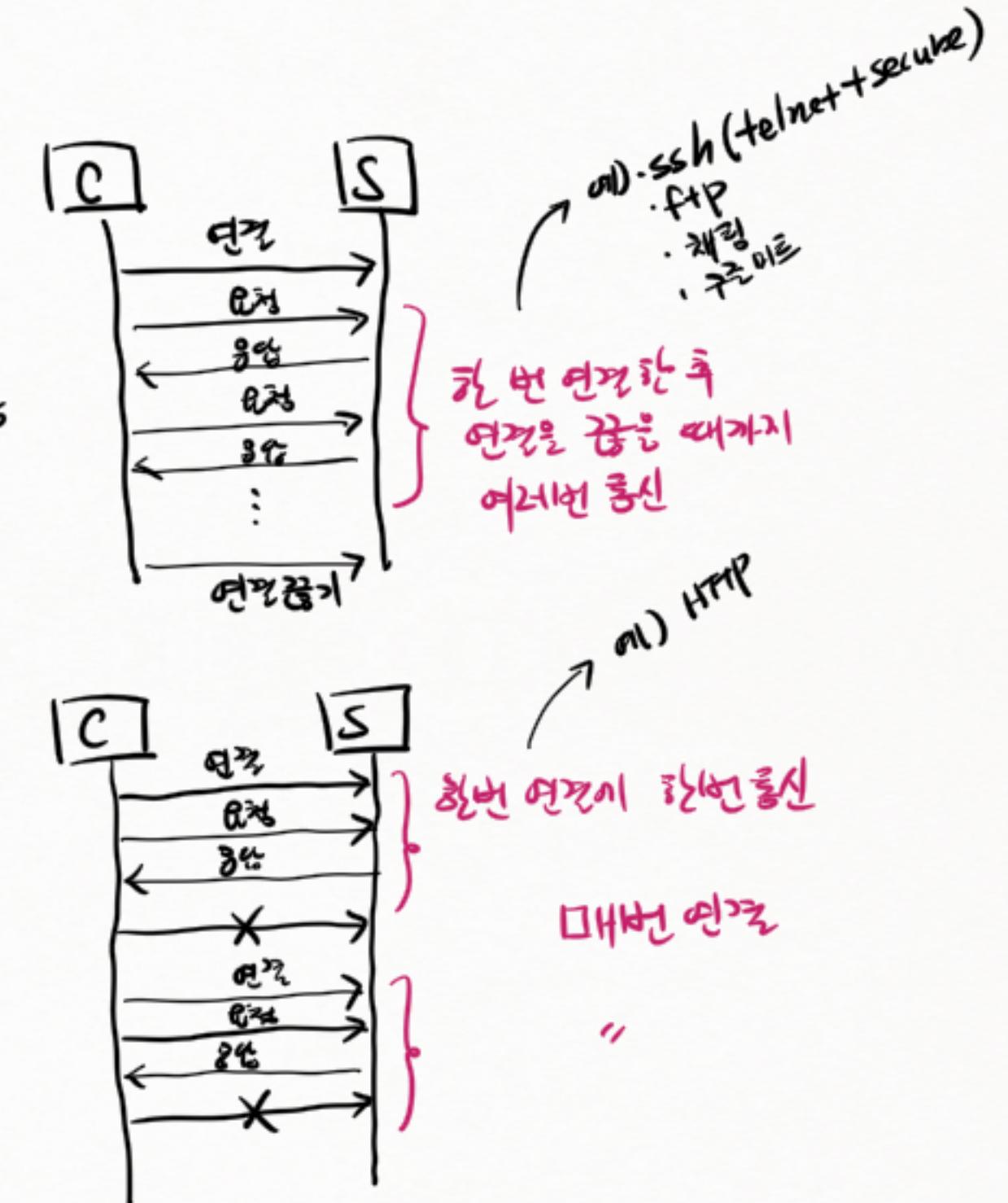
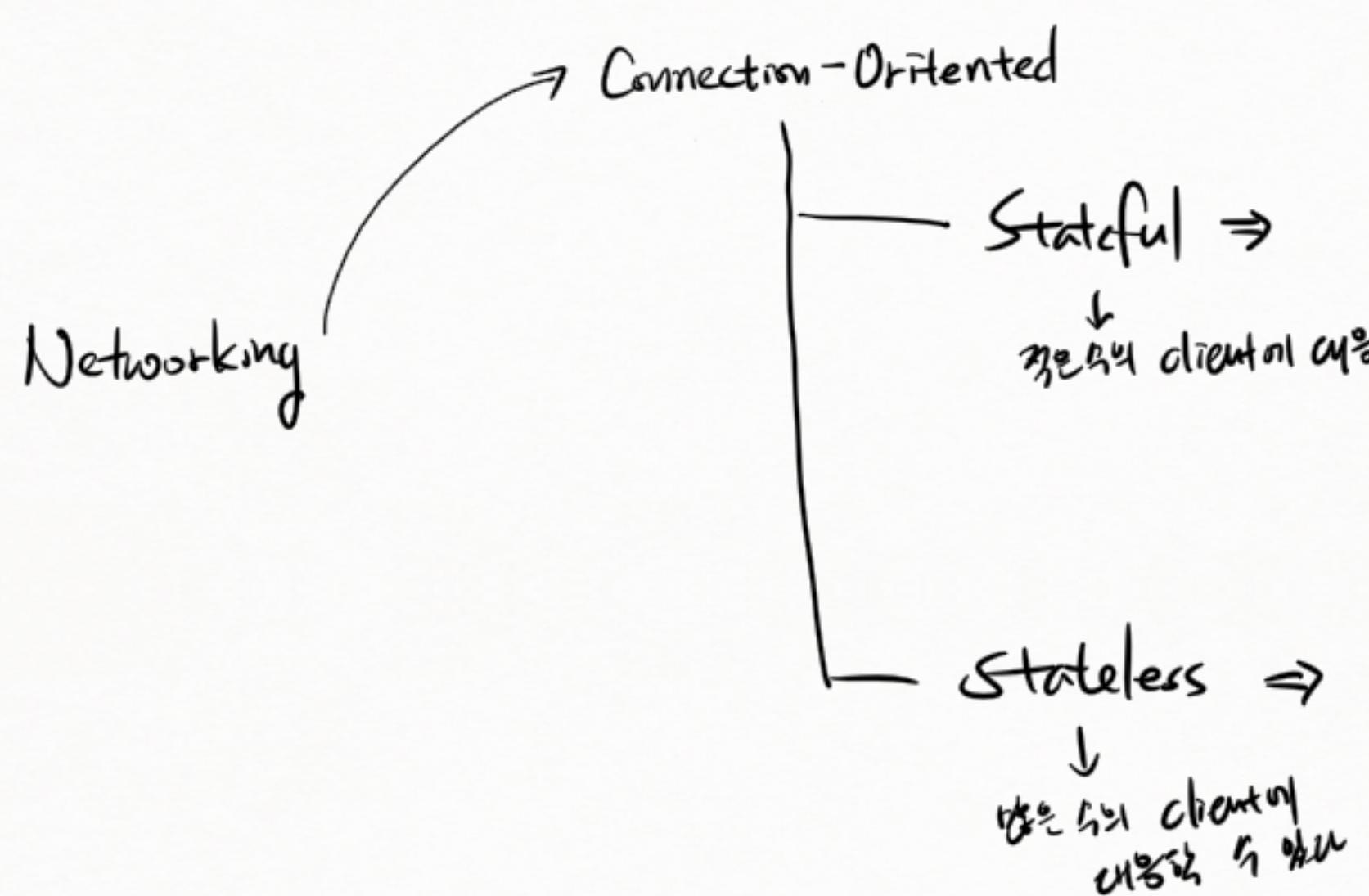
파일

파일 확장자는 Web 등 여러 가지 확장자로

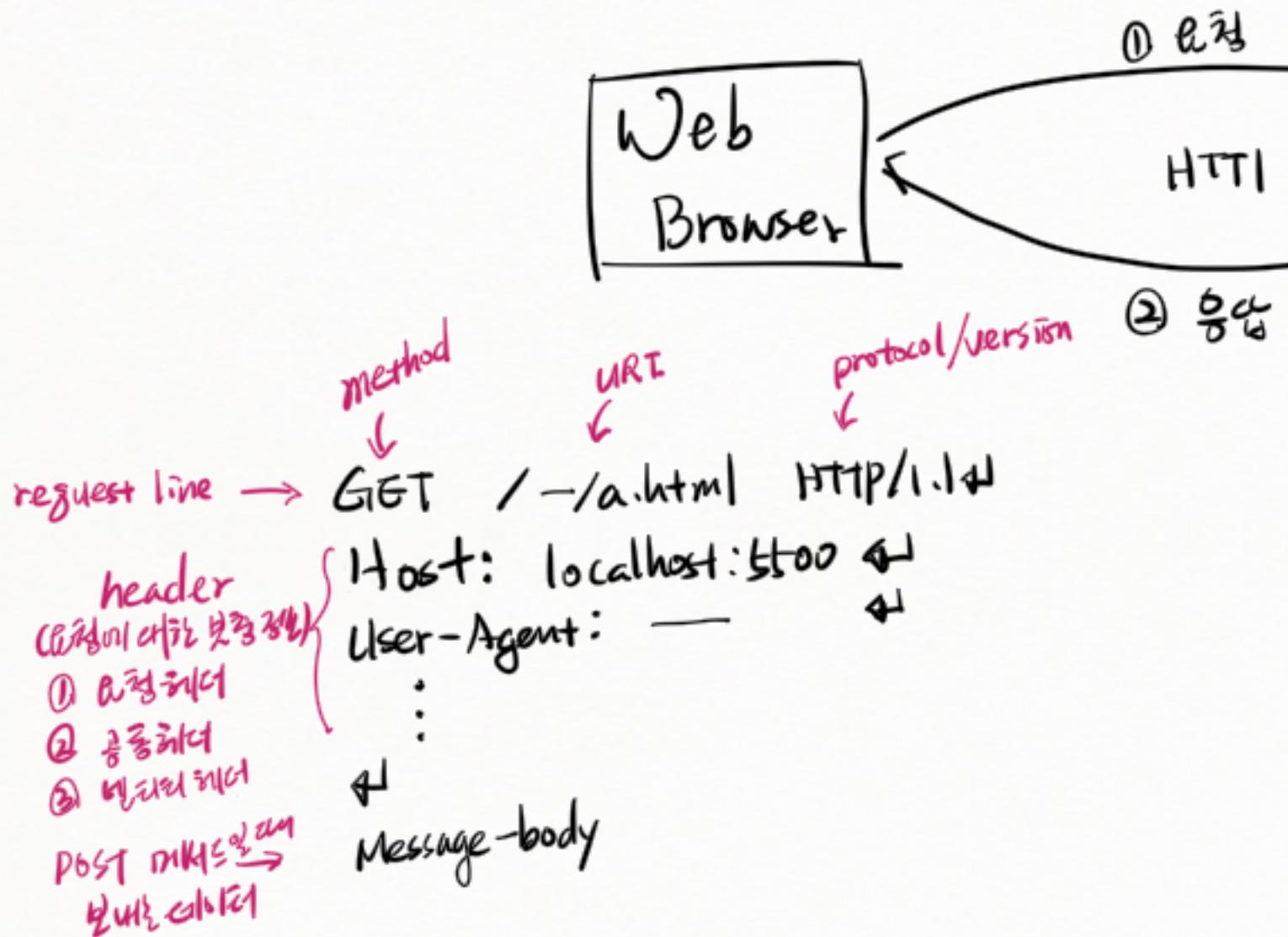
HTTP



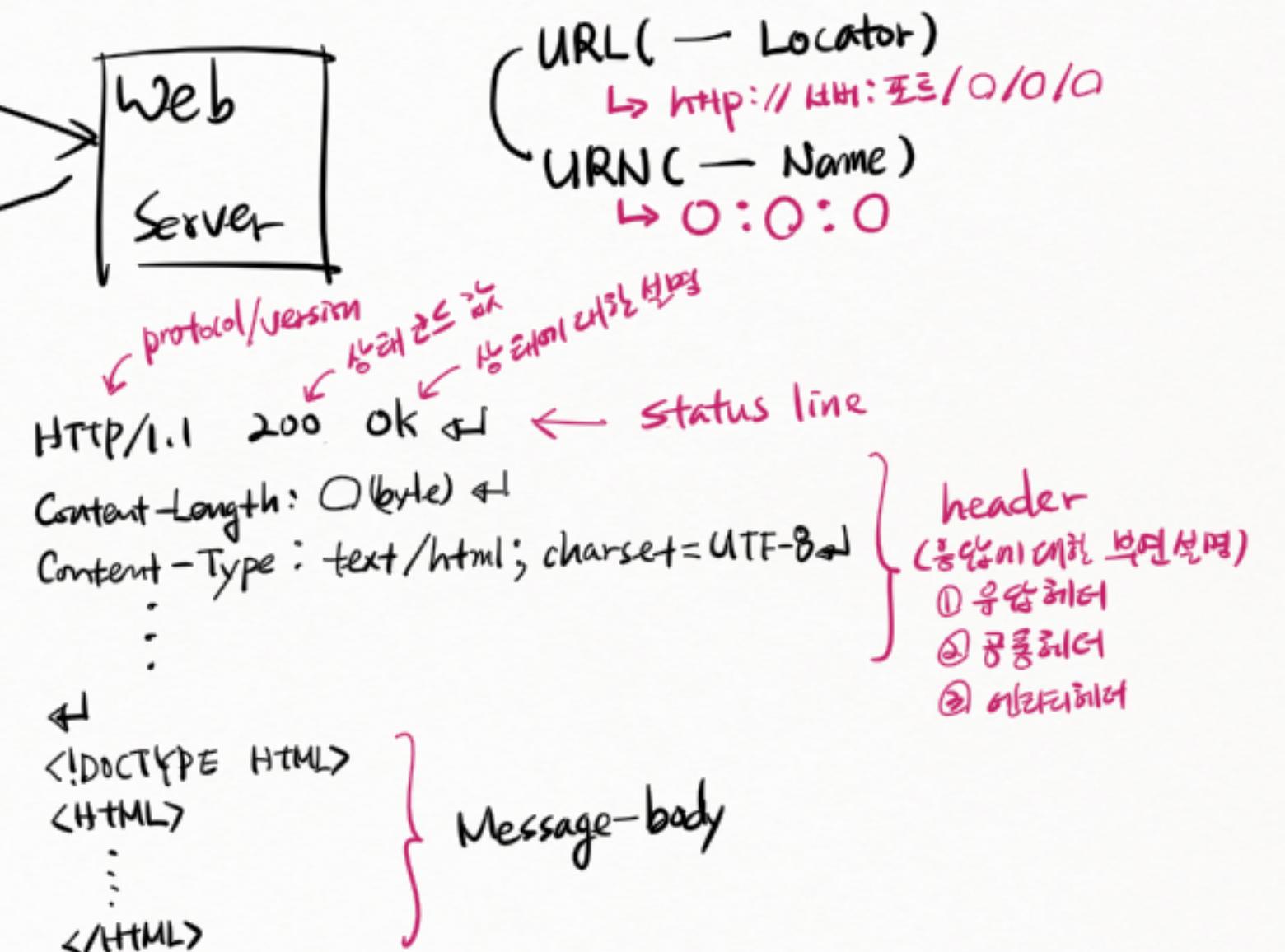
* 네트워킹



* HTTP 요청 / 응답

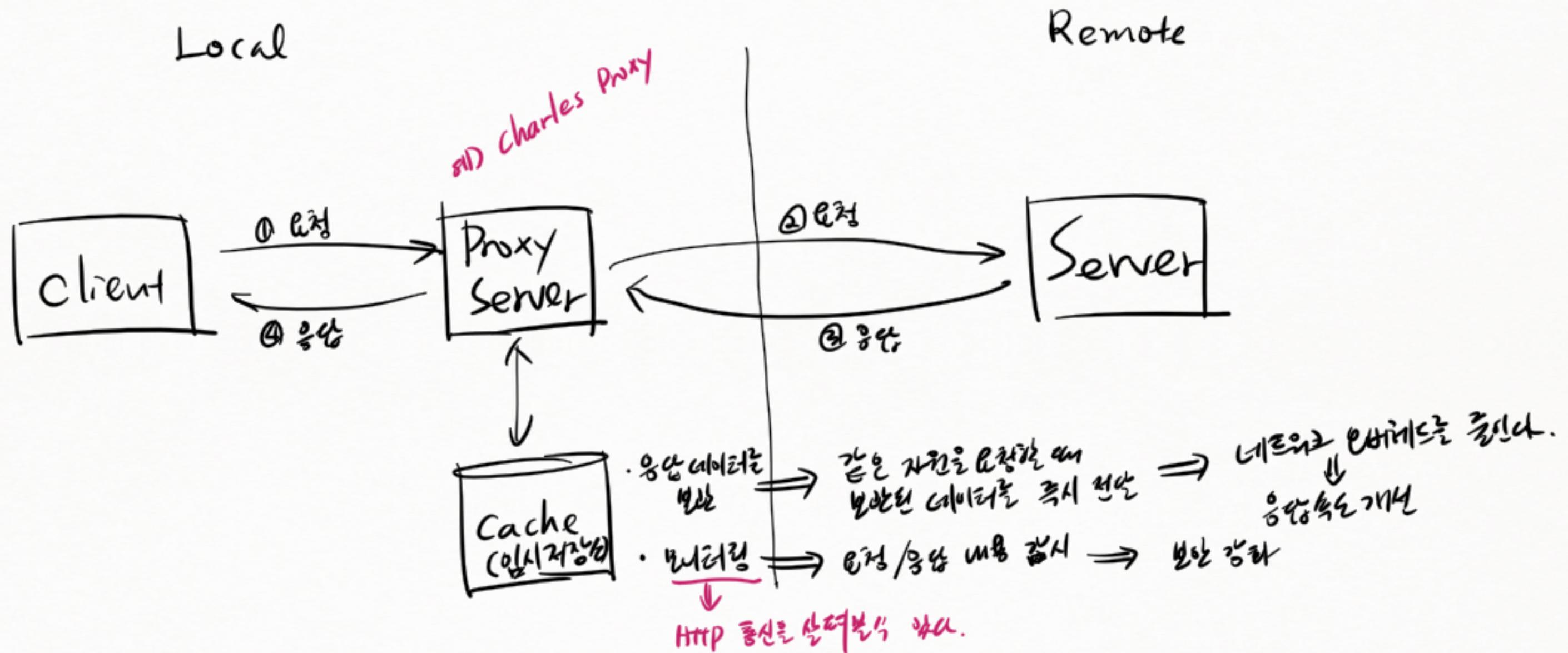


* URI (Uniform Resource Identifier)

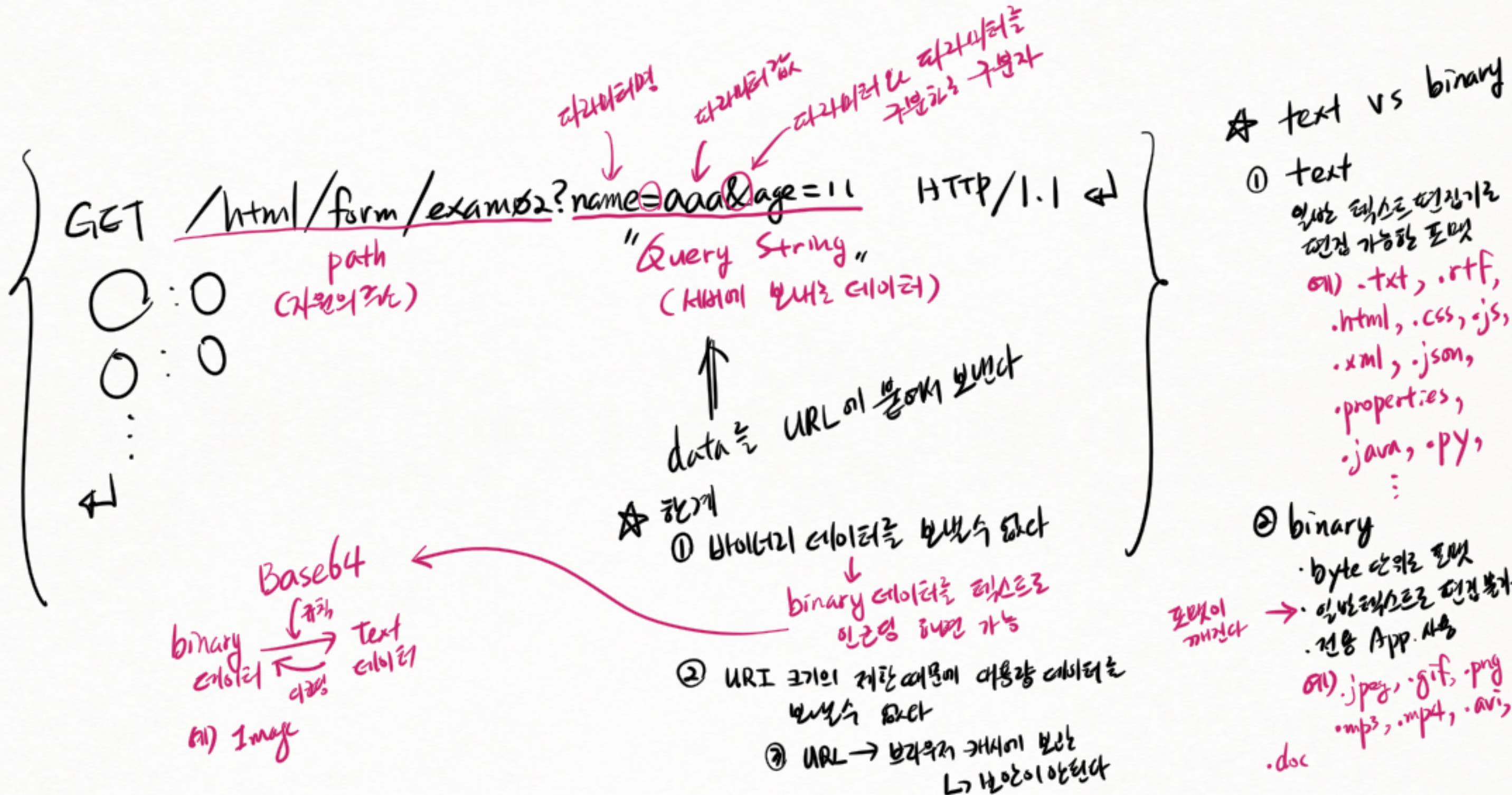


* Proxy HTTP

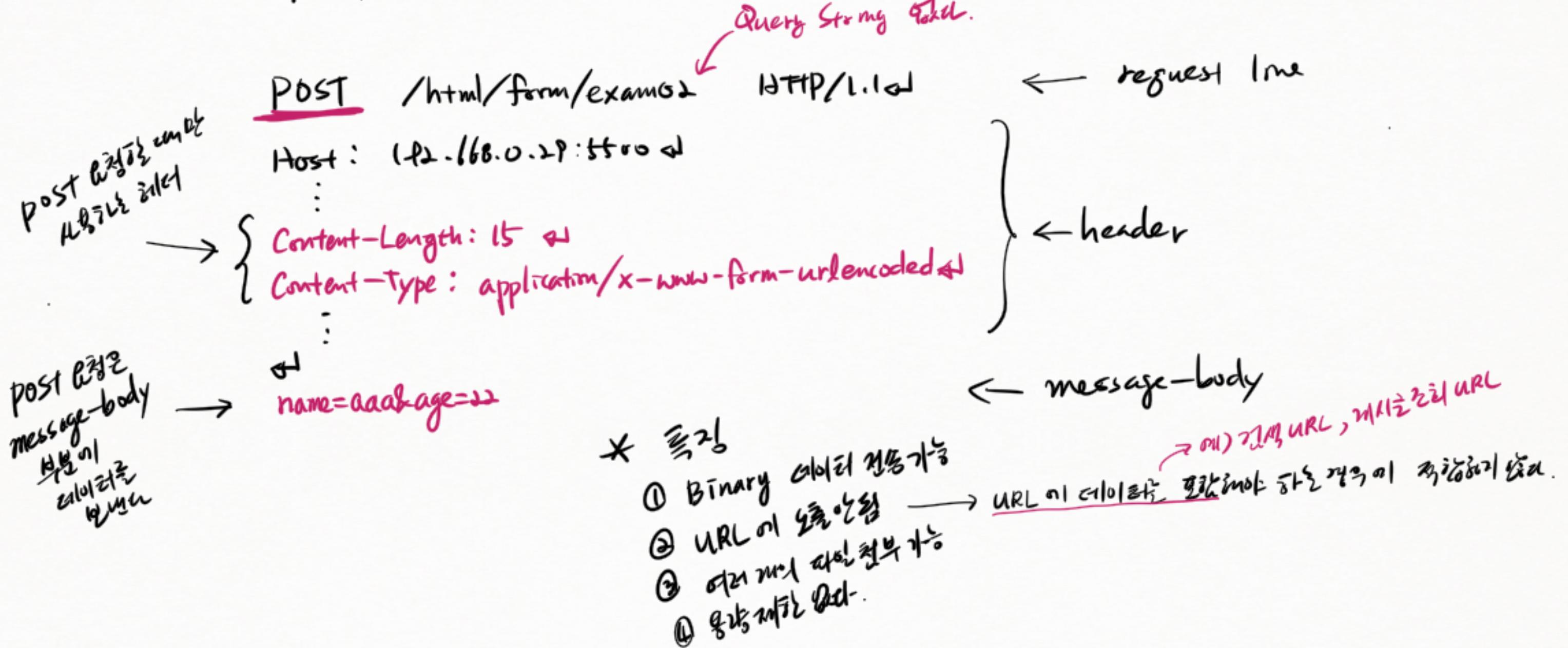
↑ client et Server 간에서 통신을 증가



* GET 요청



* POST 펴보기



* name 속성

```
<input type="text" name="title">
```



? title = ABC

자바스크립트

자바스크립트

자바스크립트

* name 키워드 II

<input type="text" name="name">
<input type="tel" name="phone">

abc

010-1111-2222



? name=abc & phone=010-1111-2222

css

* CSS : HTML element의 모양을 정의

↳ Cascading Style Sheet

selector

스마일 적용할 대상을
가리킨다.

.content header > span.title:hover {

background-color: red;

color: white;

}

↑
style name

↑ style value

pseudo selector
(선택자의 형태)

⇒ 셀렉터 { 스마일: 값; 스마일: 값; ... }

☆ ① selector 히어러

☆ ② specificity (스마일 적용순서)

③ box 다루는 방법
(레두리, 여백, 박스 레이아웃)

④ 폰트 다루는 방법

⑤ 색상 지정하는 방법

⑥ 배경 다루는 방법

⑦ block or inline 다루는 방법

⑧ 위치 조정하는 방법



* Selector

* {} — }

* {} — }

태그명 {} — }

body {} — }

태그명, 태그명, 태그명 {} — }

img, ul {} — }

.2중명 {} — }
class

.c1, .c2 {} — }

#태그아이디 {} — }

#content {} — }

-[속성명=값] {} — }

-:상태명 {} — }
pseudo selector

li:hover {} — }

진상 자손 {} — }

#content li {} — }

부모 > 자식 {} — }

ul > li {} — }

태그 + 다음태그 {} — }

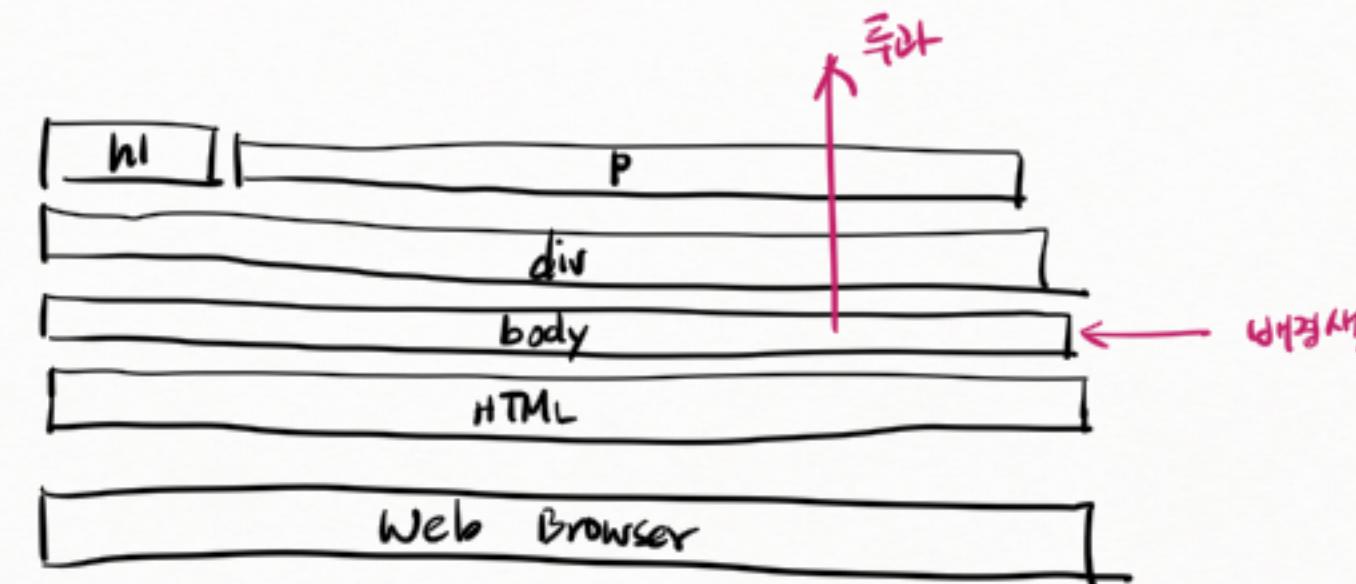
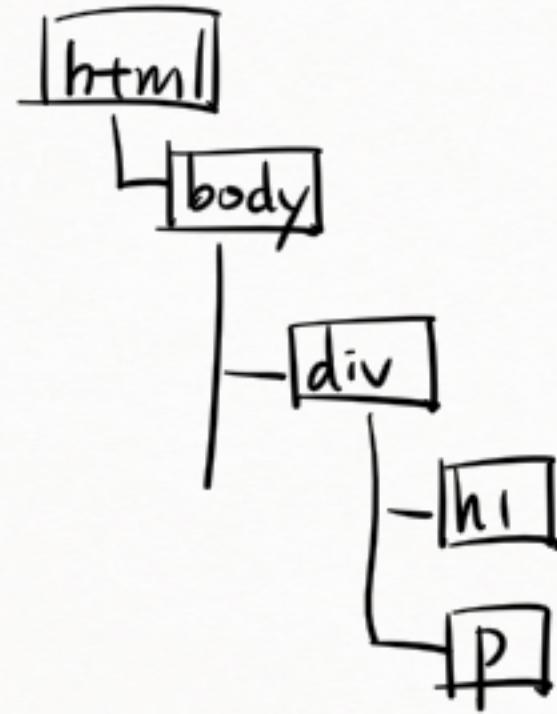
img + ul {} — }

대상자 조건 값 ... {} — }

div#menu li:c2 {} — }

input[type="text"] {} — }

* 부록 태그와 자식 태그의 위치 찾기



* CSS specificity : 스타일 적용 순서 낮 → 높

* → 0

id, pseudo selector → 1

class, tag → 10

attribute → 100

inline style → 1000

* {}

h1 {} a:hover {}

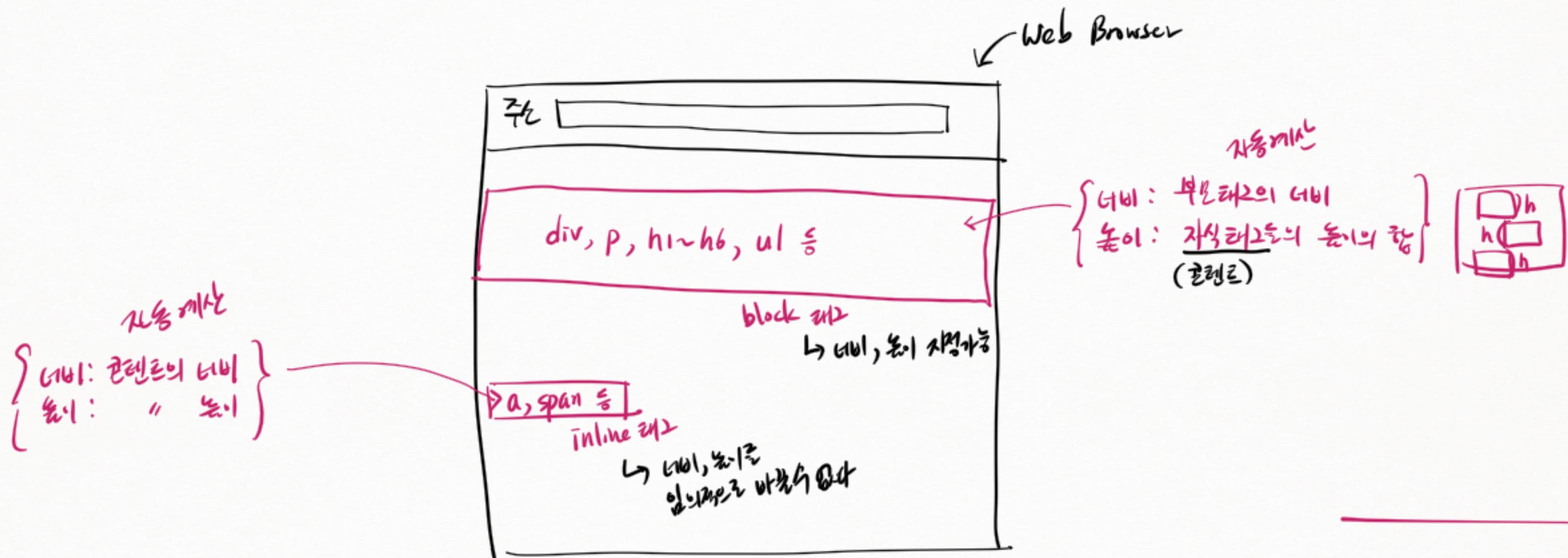
.cl {} input[readonly] {}

#menu {}

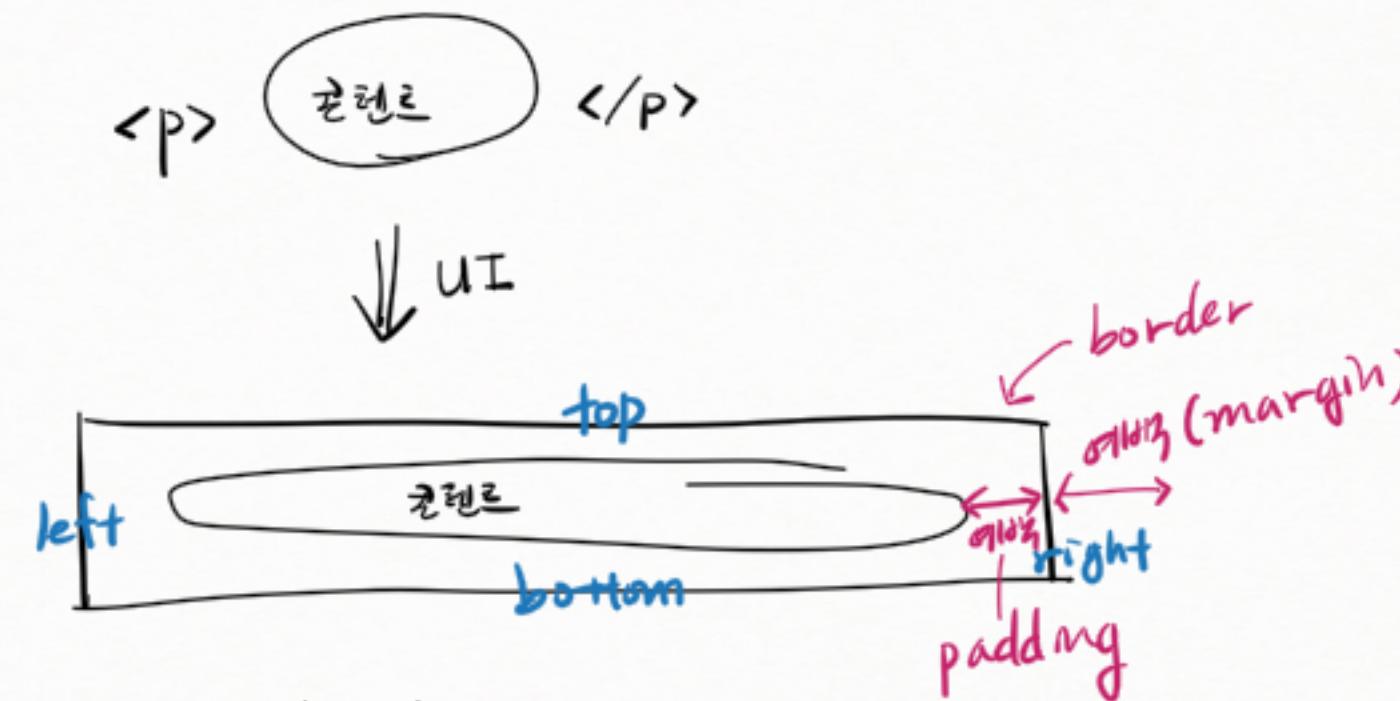
 ()

인라인 스타일

* 태그의 너비, 높이



* 레이아웃 (border)



right
bottom color dotted
border-top-style: solid;
left width dashed
:

짧은 명령
border-top: color style width;
border-bottom:
border-right:
border-left:

border-style: solid;
모든 방향의 레이아웃 스타일

더 짧은 명령
border: color style width;

레이아웃 방향

* Raster 폰트 Vector 폰트

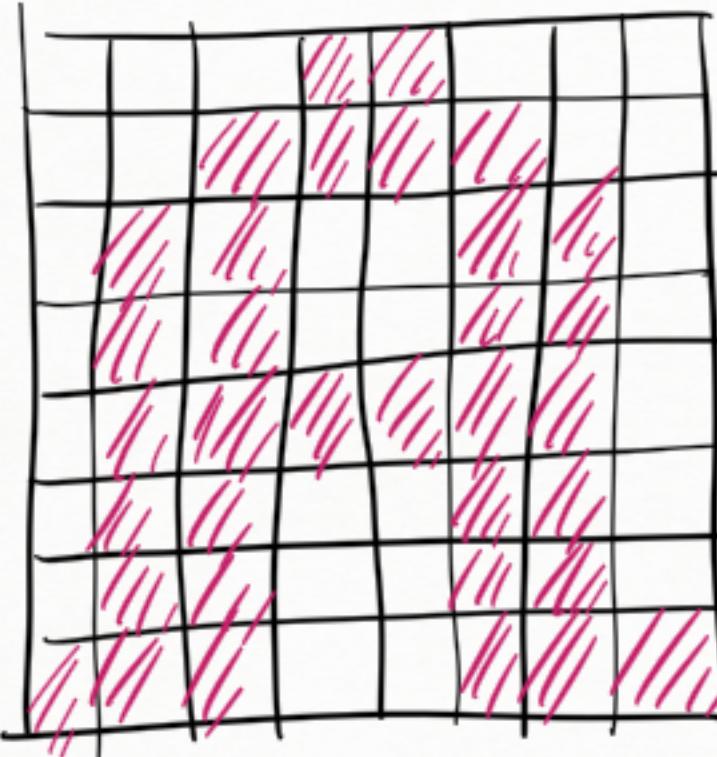
↳ 비트맵 이미지 예) .bmp, .gif, .png, .jpg ...

↳ 벡터이미지 예) 층집마크, 캐드

Raster 폰트

↳ 퍼센트 단위로 글자를 만든다

- 한글 속도가 빠르다
- 폰트크기에서 따라
각 문자를 만든다
- 정해진 크기보다
더 크게 한글 속도를
얻을 때 각 문자의
크기를 늘리기 때문이다
제작 현상 발생
- 이미지의 불확정화
상관없이
파일크기는 같다.



(1) COURIER

Vector 폰트

- 글자를 그리는 명령어를 작성한다
- 글자를 그리는 명령을 수행

* 폰트크기 = 높이

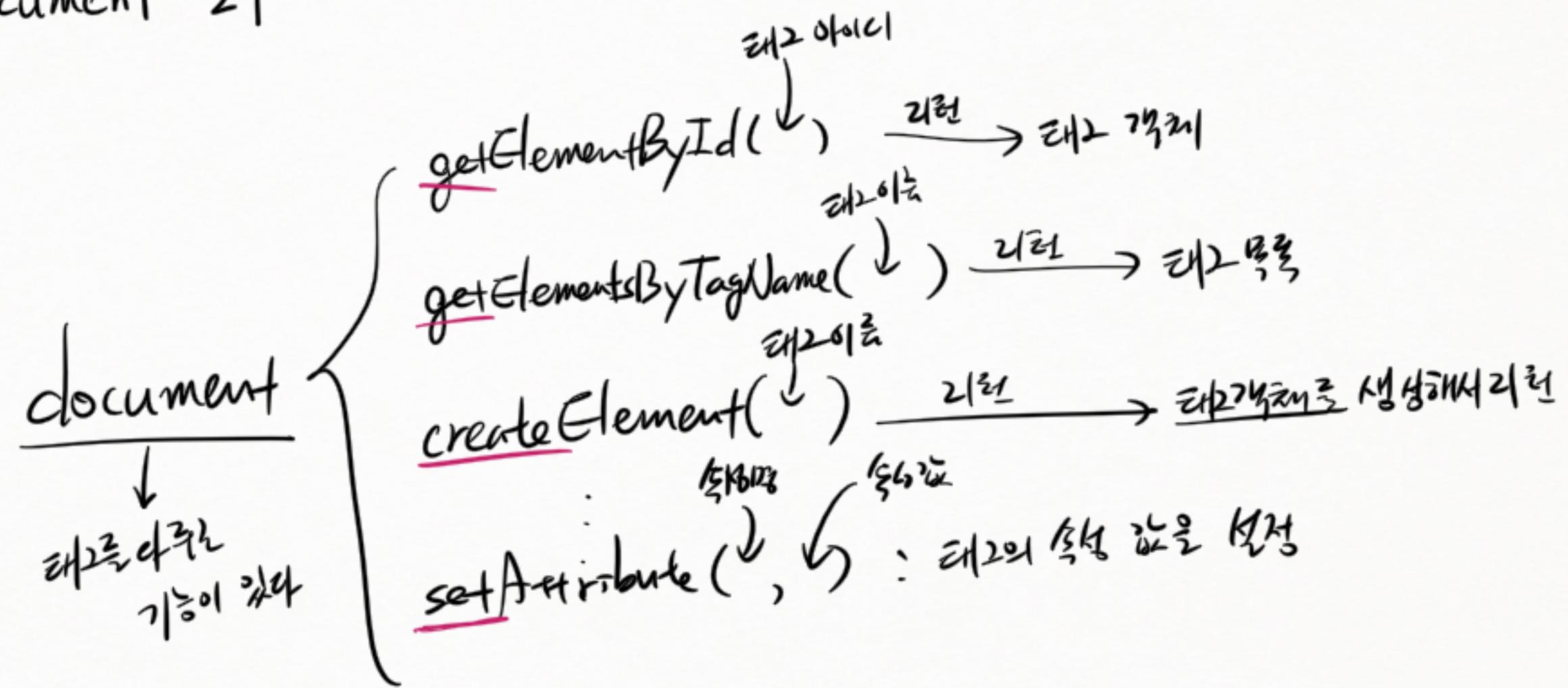
4~5 line
3~6 line
(3,3) (1,8) line
⋮



(1) True-Type 폰트
courier new

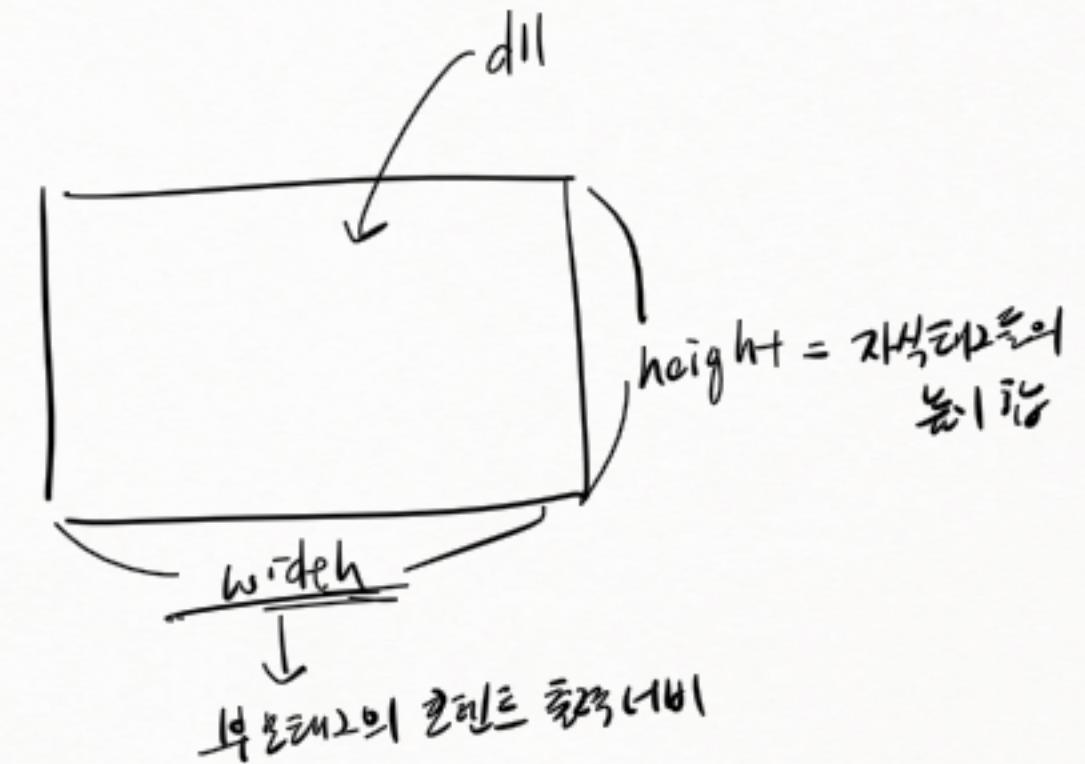
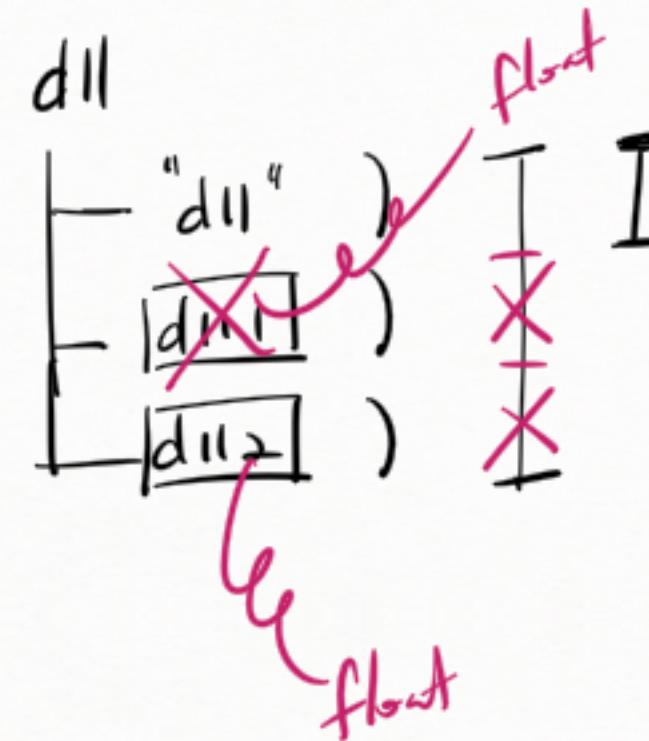
- 한글 속도가 느린다
(같은 CPU가 빠르기 때문에
상관하지 않는다)
- 글자 크기를 늘리더라도
명령을 통해 그리기 때문에
제작 현상이 발생하지
않는다.
- 이미지 크기 → 파일크기 ↓
" 확장 → " ↑

* document 속성

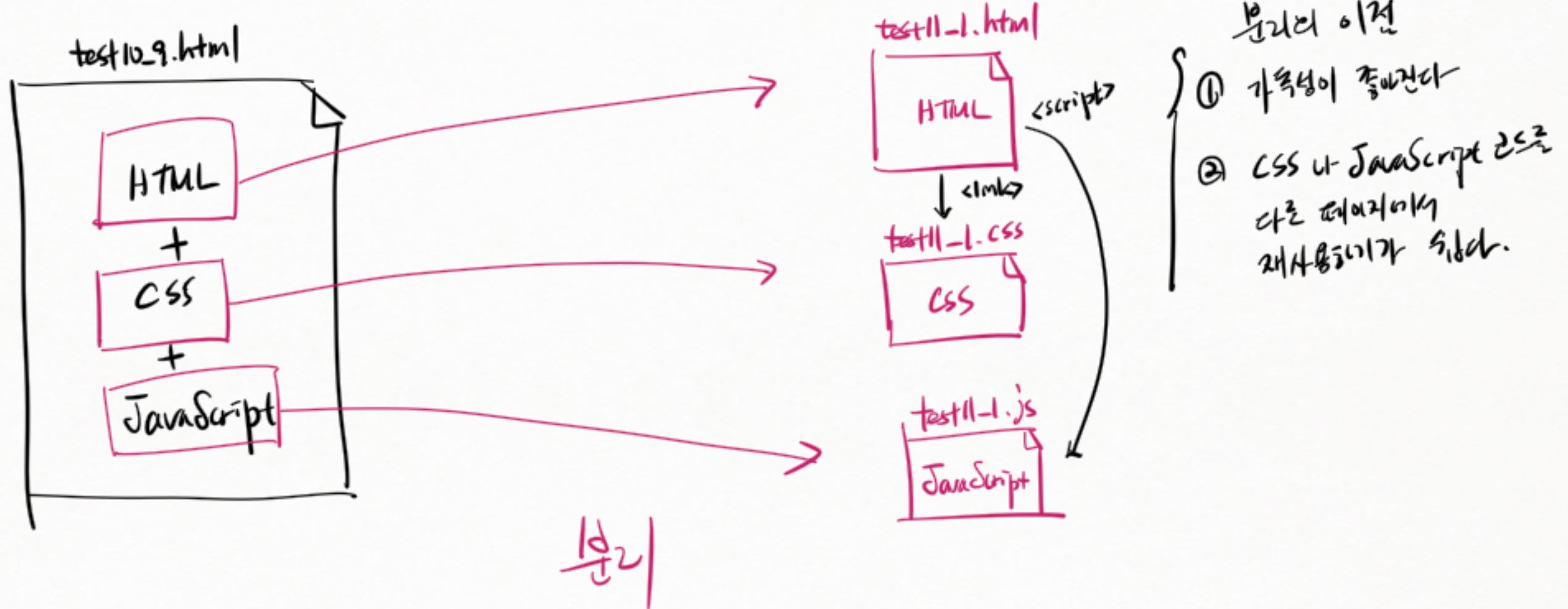


* float 2f absolute

↳ 부모태그의 크기를 계산할 때 float는 absolute 태그를 제외한다.

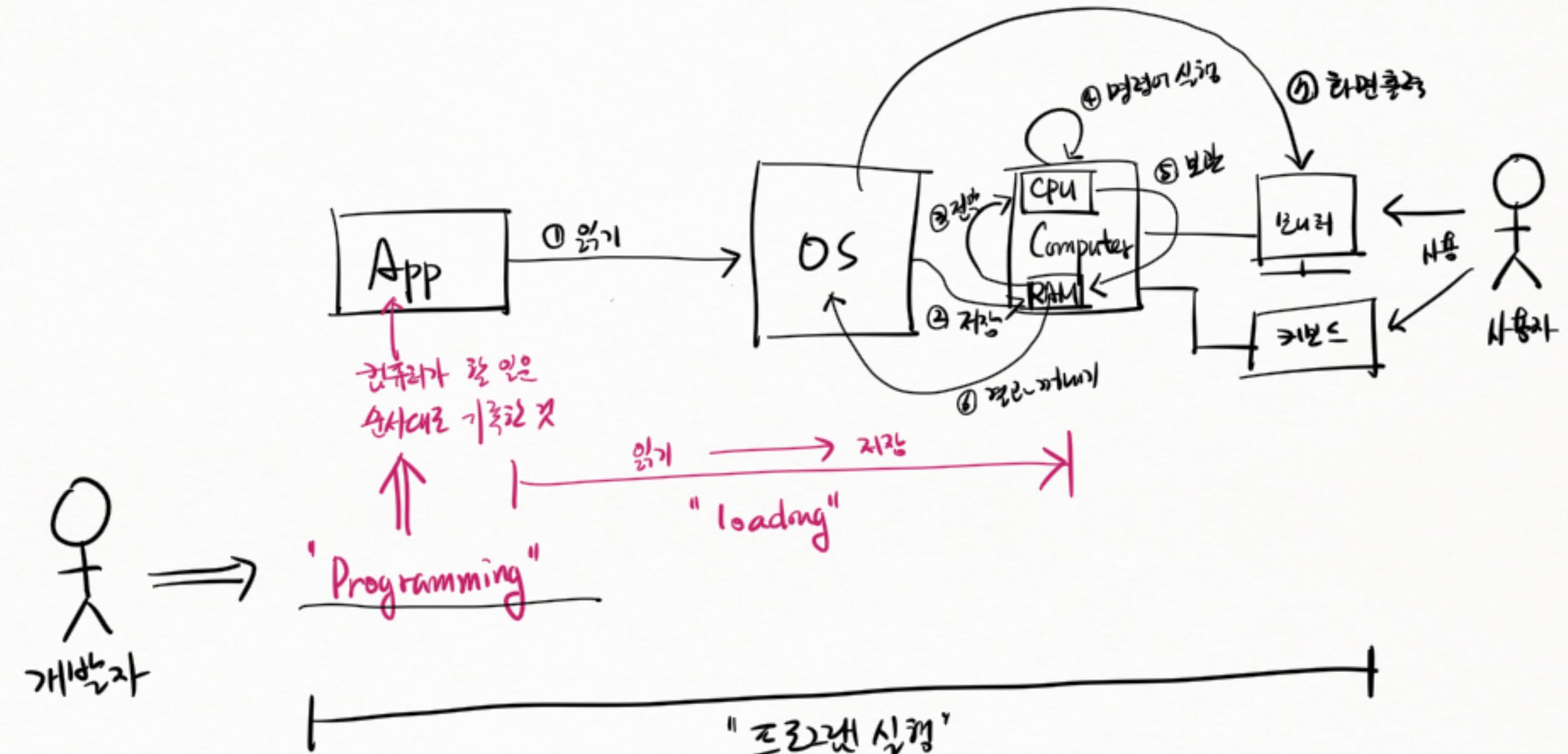


* CSS와 JavaScript 코드의 분리

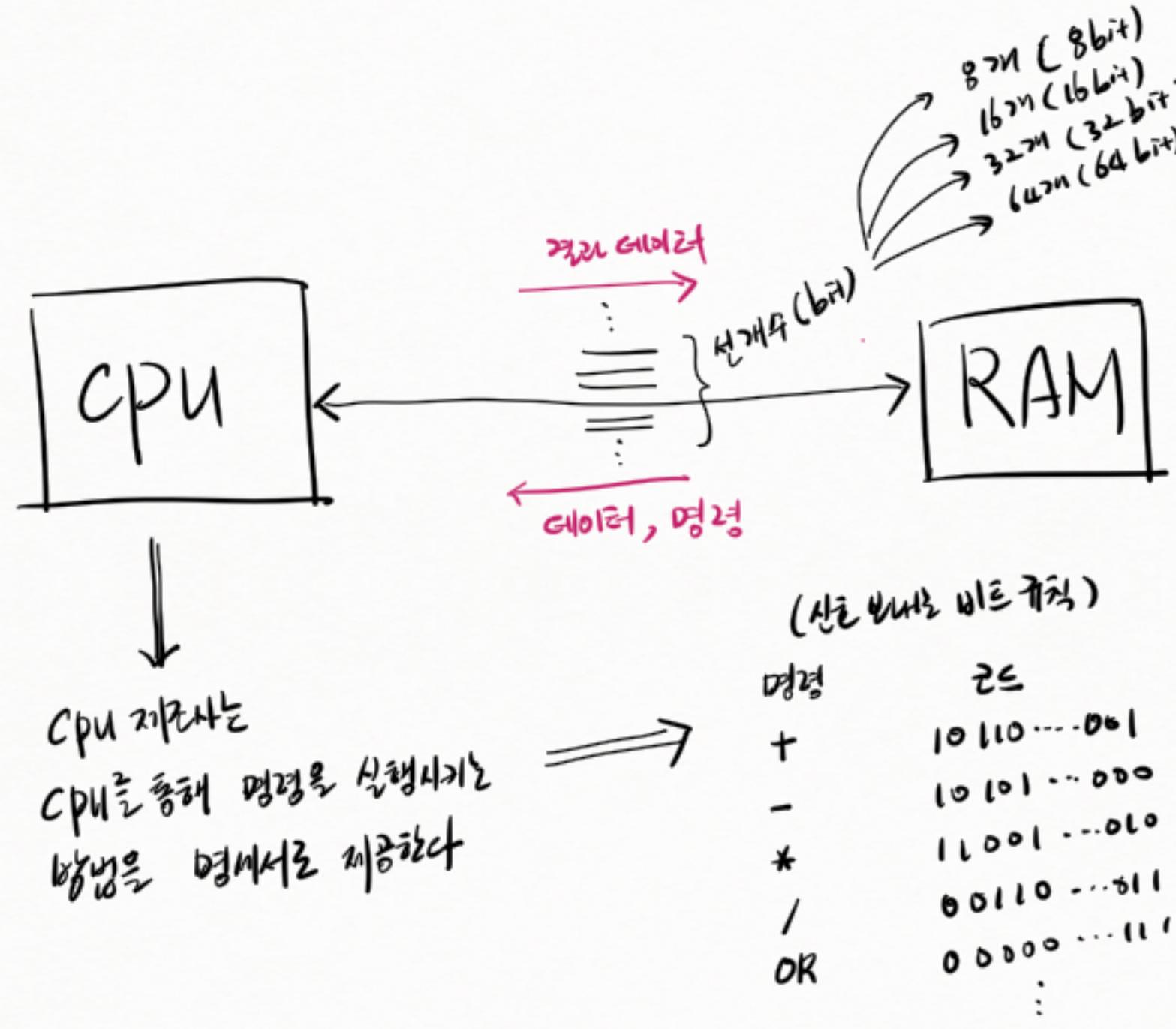


JavaScript

* 프로그램 실행과 프로그래밍



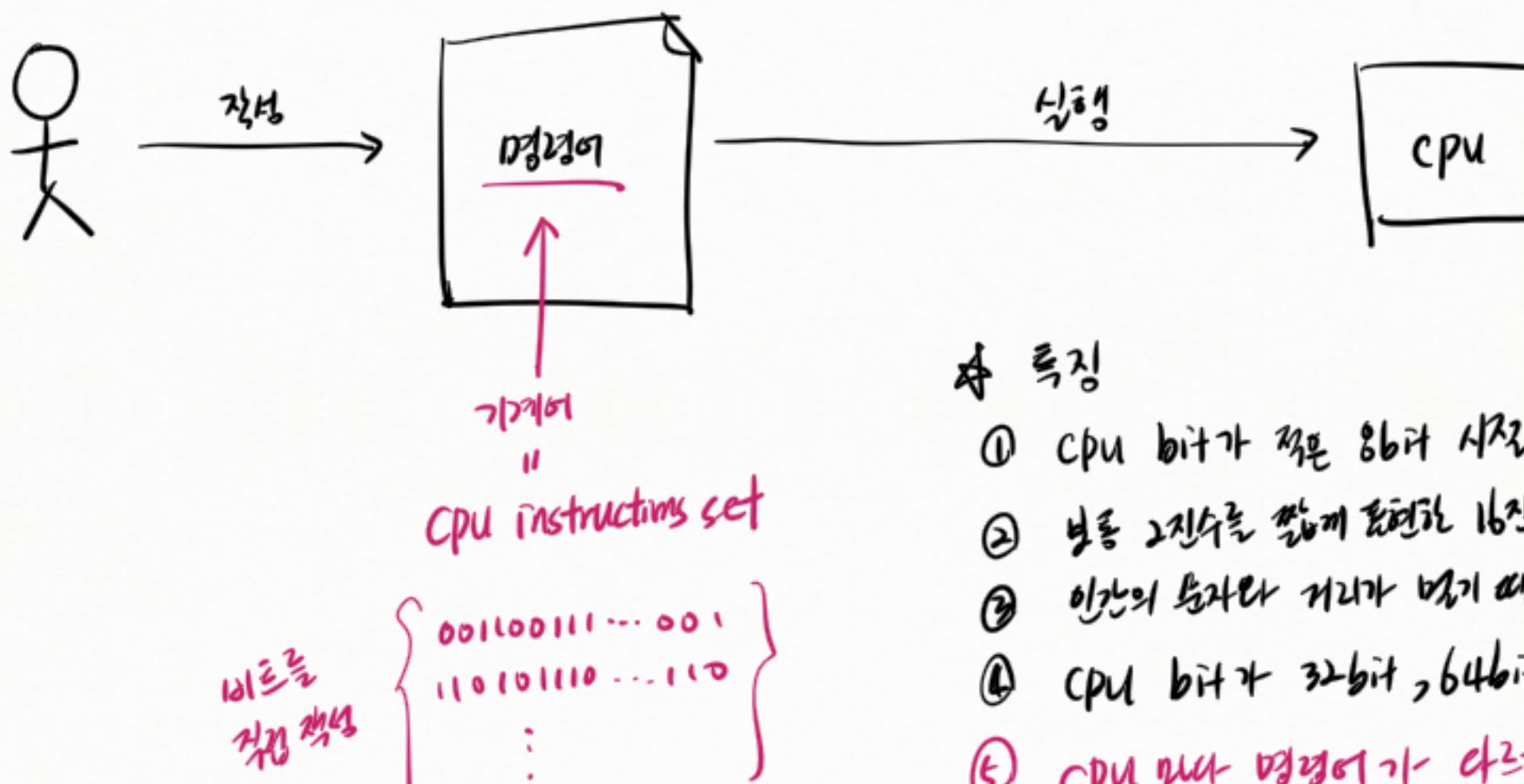
* CPU 와 RAM , bit



$$\begin{array}{r}
 1 \ 1 \ 1 \ 1 \ 0 \ 1 \\
 0 \ 0 \ 0 \ 0 \ 1 \ 0 \quad + \\
 1 \ 1 \ 0 \ 0 \ 1 \ 1 \quad - \\
 1 \ 0 \ 0 \ 0 \ 1 \ 0 \quad *
 \end{array}$$

* 명령어 작성 : 기계어

① CPU instructionset 명세서를 보고 직접 명령어 작성

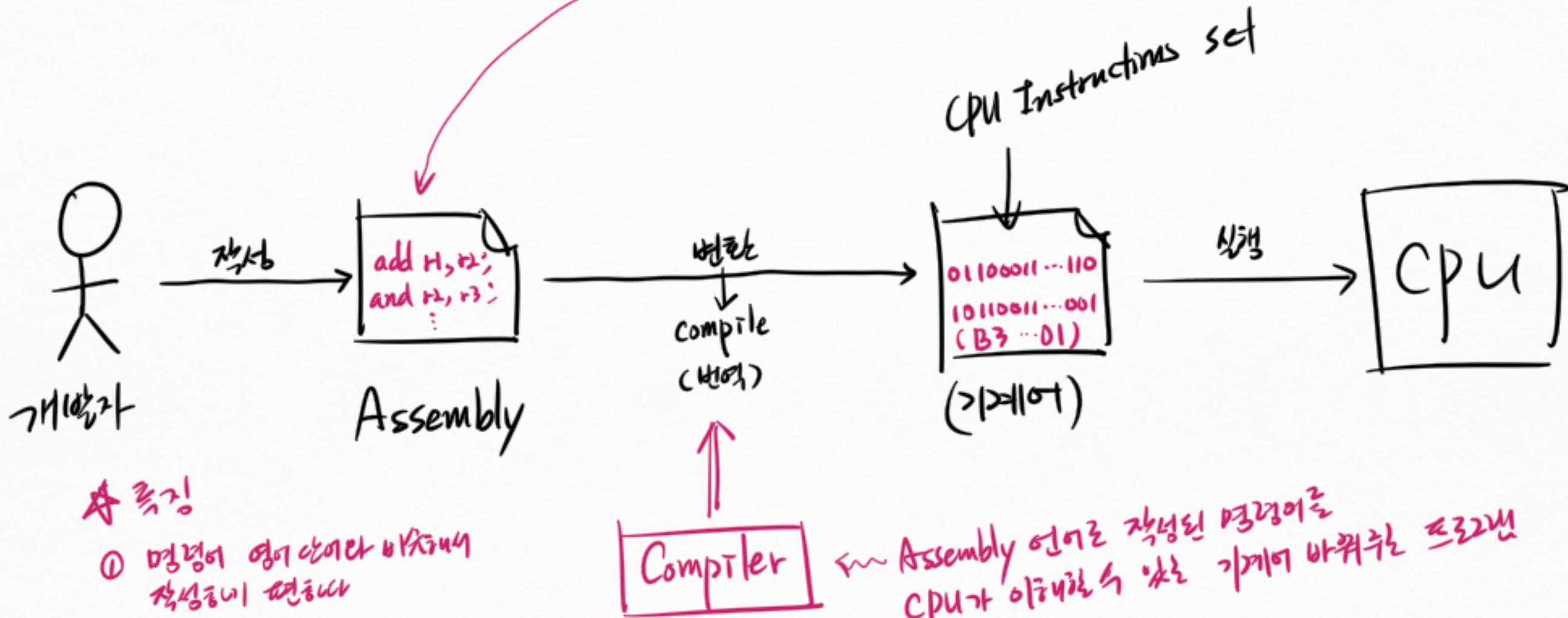


* 특징

- ① CPU bit가 적은 8bit 시장에는 개발자가 직접 작성하기도 한다
- ② 보통 2진수를 끌어와 표현한 16진수를 사용해서 작성한다
- ③ 이진의 둘자리 거리가 멀기 때문에 작성하기 매우 불편하고 힘들다.
- ④ CPU bit가 32bit, 64bit로 늘어나면서 더 어렵게 되었다
- ⑤ CPU마다 명령어가 다르기 때문에 다양한 CPU에서 실행할 수
 전기적 신호
 비트 구조
 있는 명령어는 작성하기
 매우 힘들다.

* 명령어 작성 : Assembly

② 직장 → 기계어로 작성하는 대신 간접적 영어 문장으로 이루어진 명령어 사용



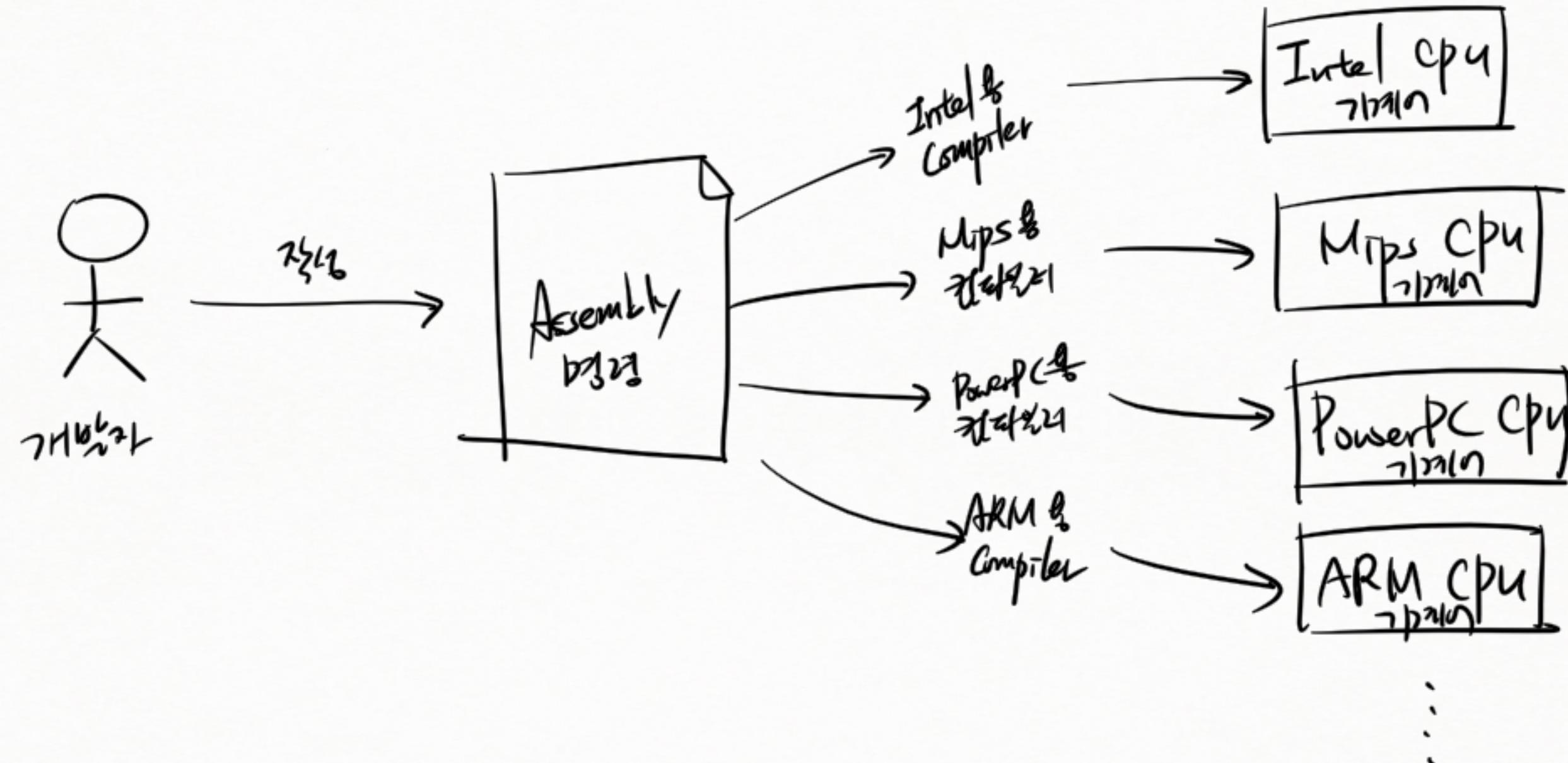
★ 특징

- ① 명령어 영어 문장과 비슷하여 작성하는데 편하다
- ② CPU마다 다르게 작성되어 헷갈리다

왜?
컴파일러가 CPU에 맞춰서
기계어로 번역해 준다.

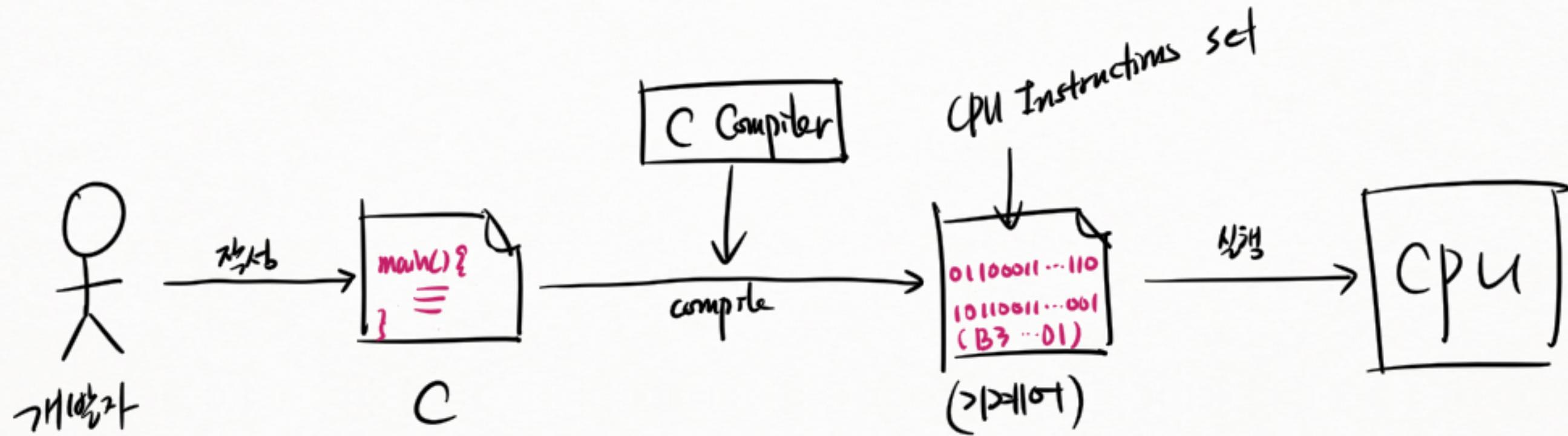
~ Assembly 언어로 작성된 명령어를
CPU가 이해할 수 있도록 기계어 바꿔주는 프로그램

* 명령어 작성 : Assembly 및 컴파일러

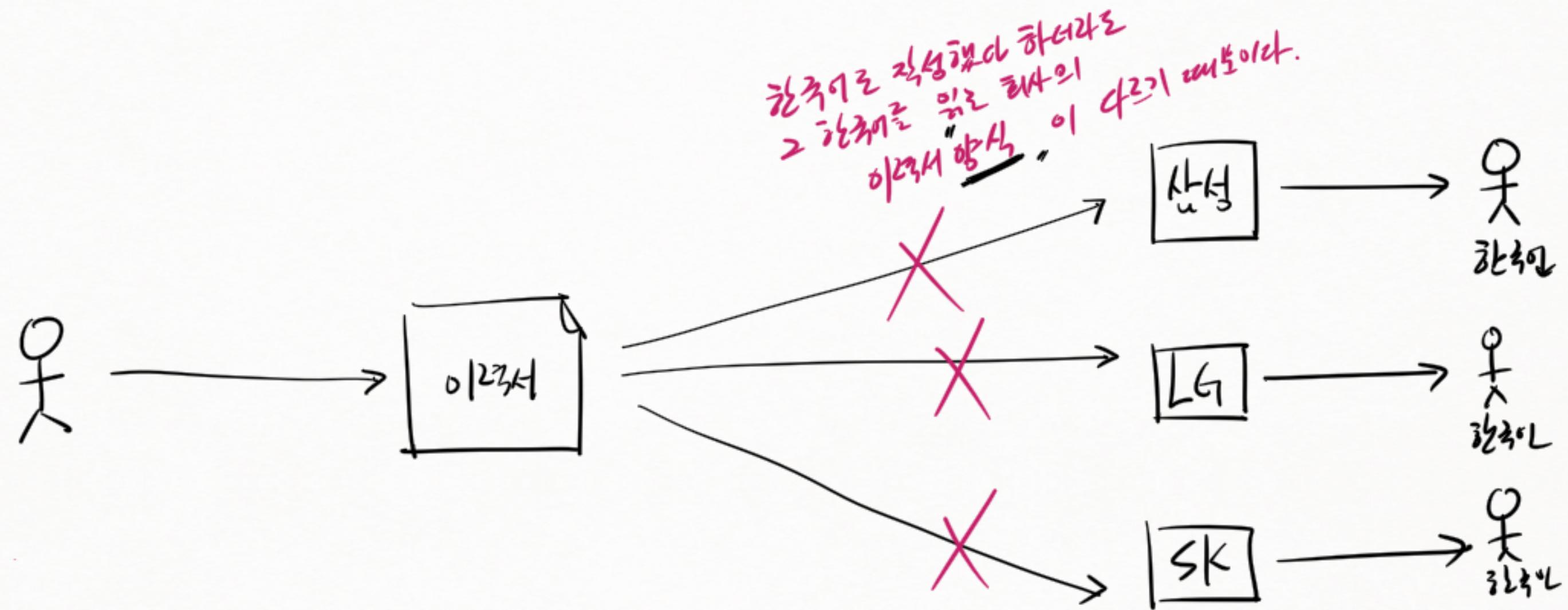


* 명령어 작성 : C

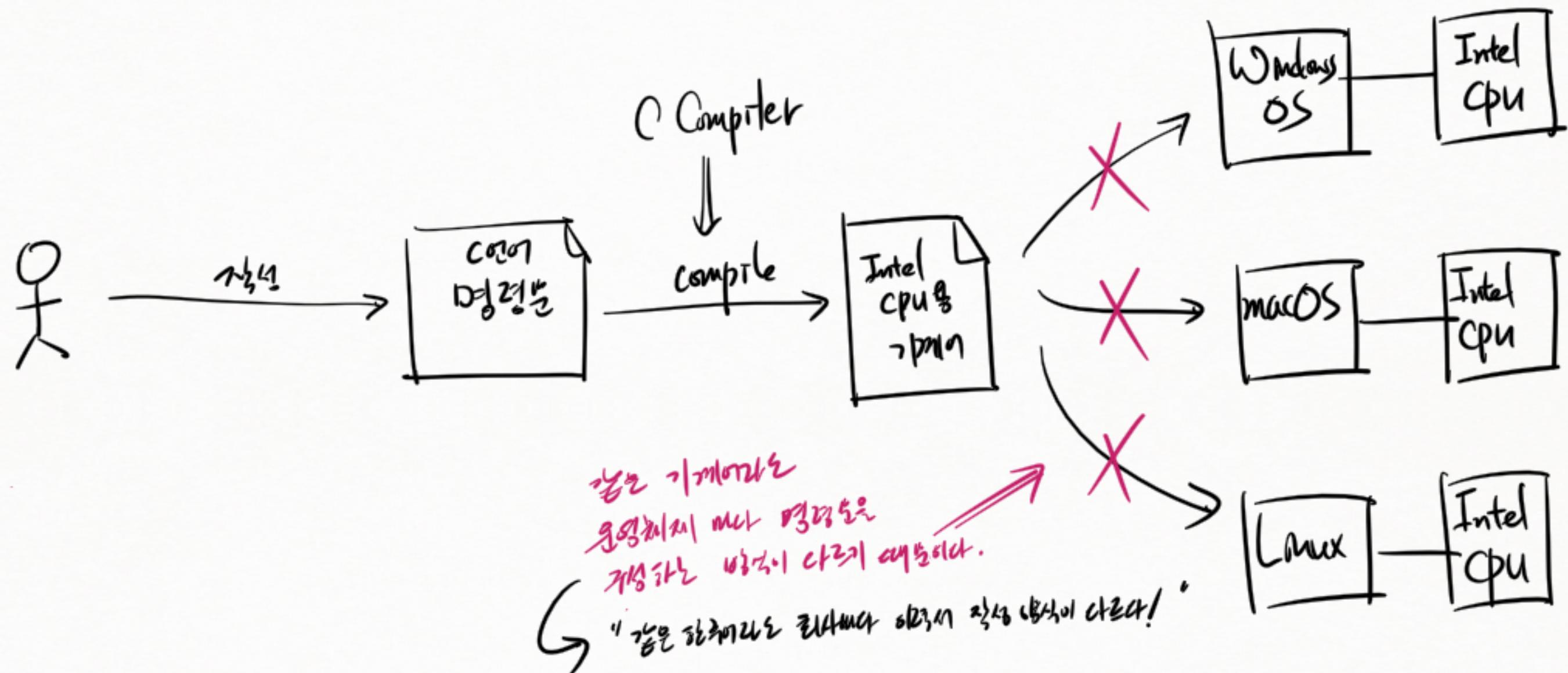
③ 더 많은 친화적인 프로그래밍 언어로 프로그램 작성하기



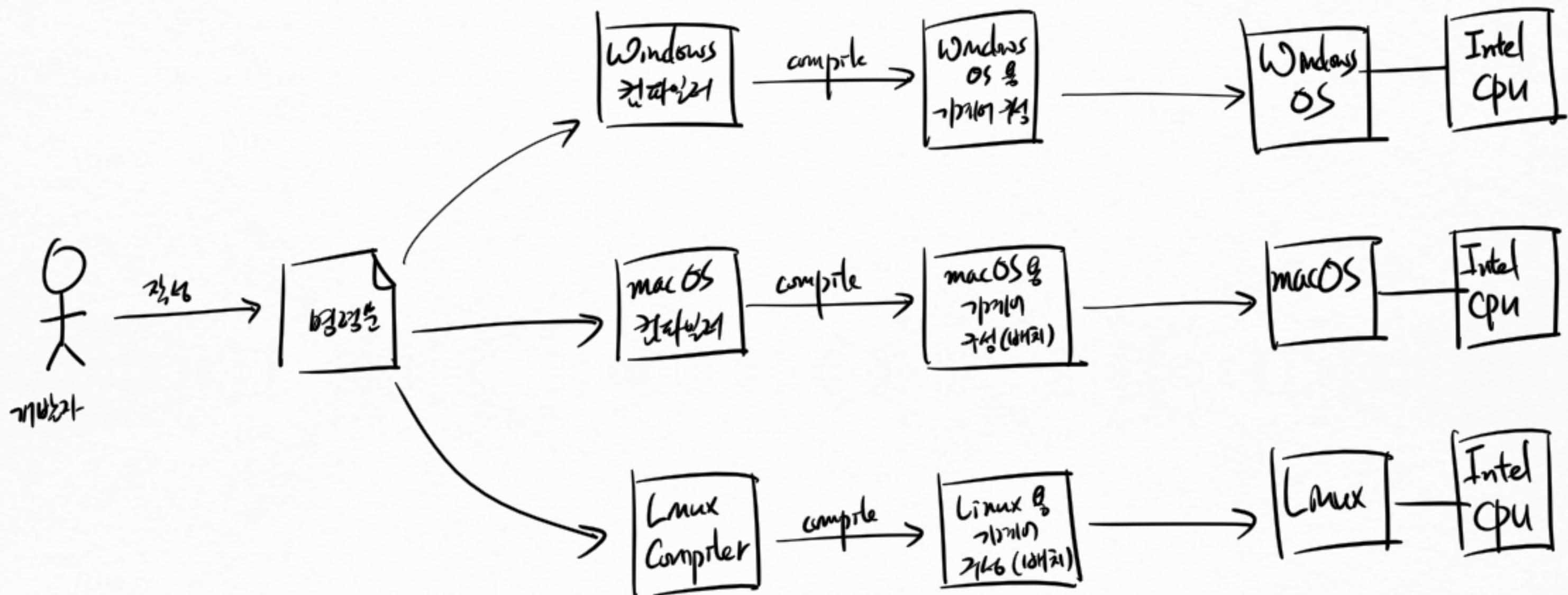
* 가기어, OS, CPU



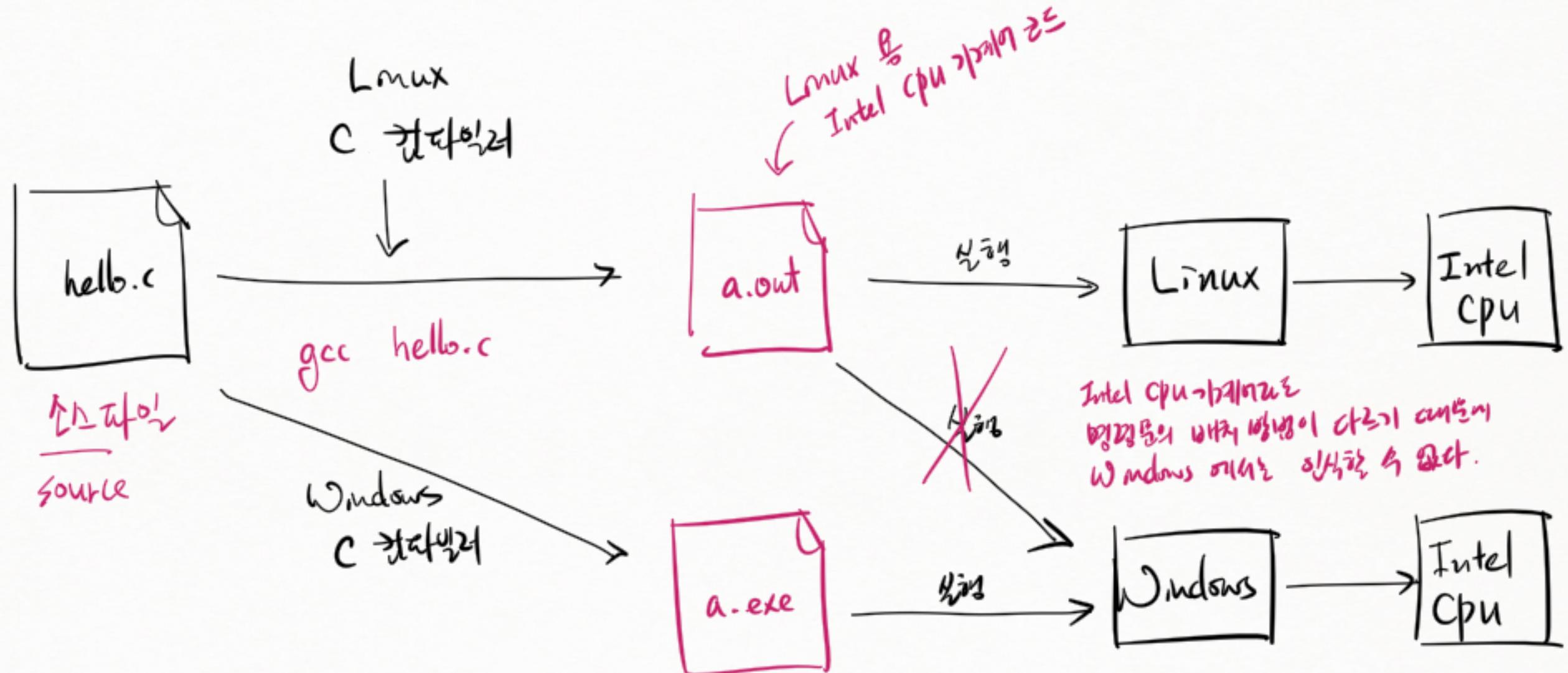
* 가상화, OS, CPU



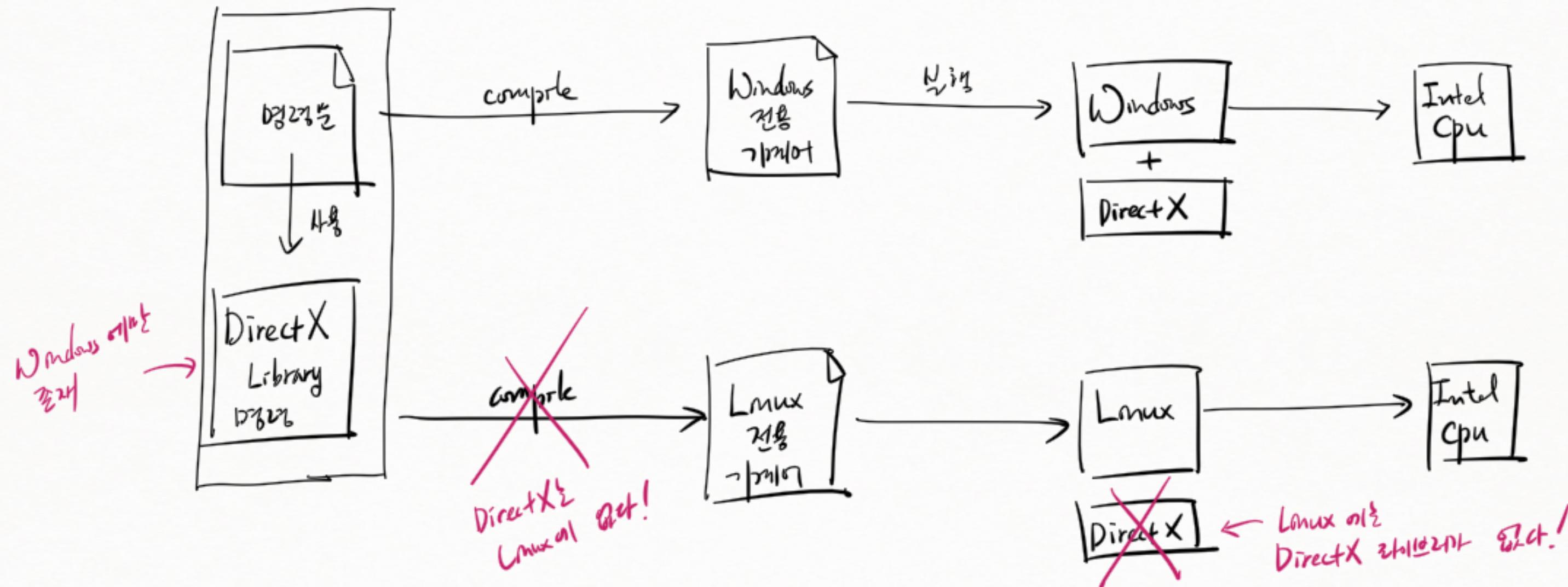
* 가기어, OS, CPU



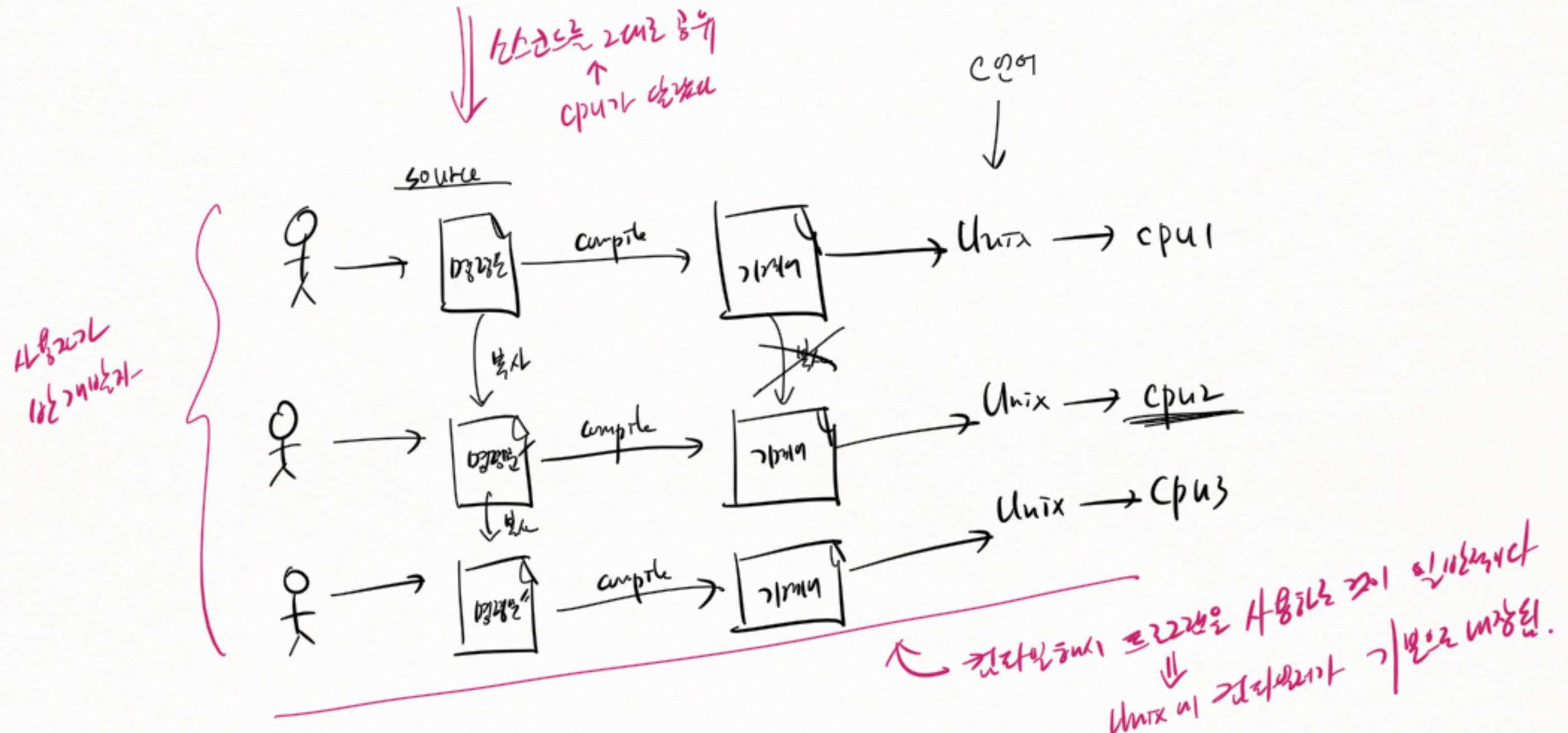
* C → 실행



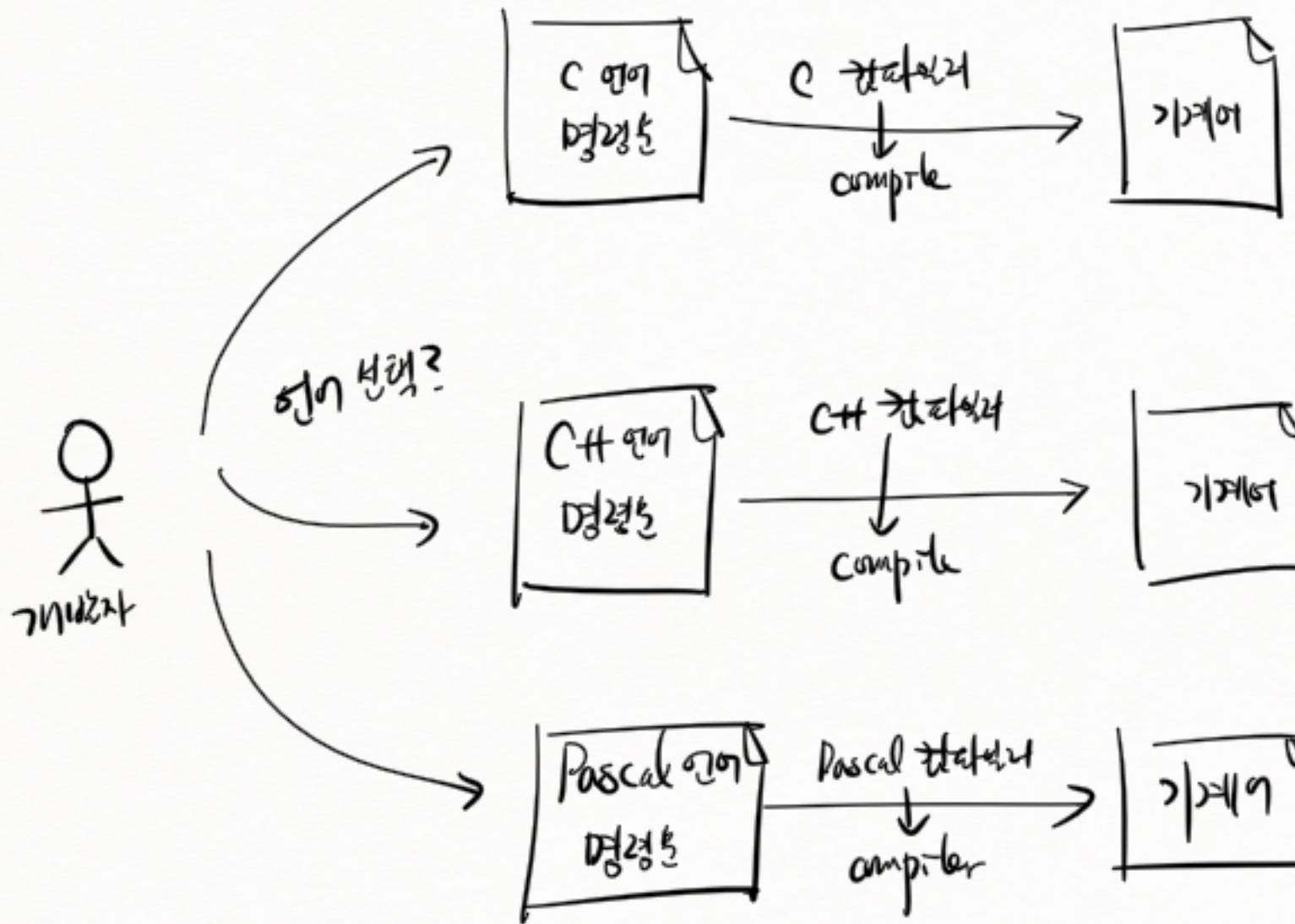
* C 언어 + OS 전용 명령 사용



* 예전의 프로그램 흐름



* 프로그래밍 언어와 → 컴파일러



* 언어 선택?

프로그래밍 언어마다 특장점이 있다



프로그래밍 언어마다 언어를 이해하는 데에

} 인공지능, 머신러닝, 딥러닝 : Python

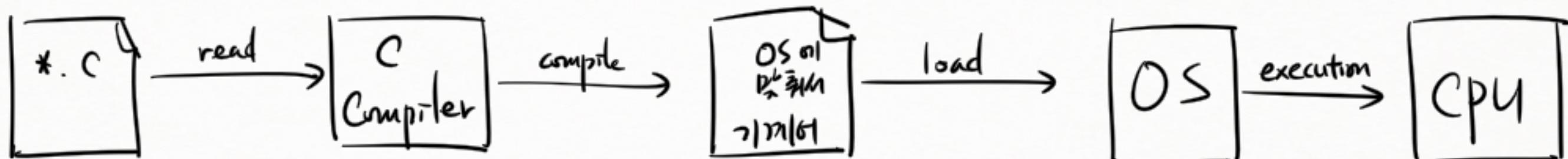
web app - : java, php, go

web UI : javascript, TypeScript

통계 : R

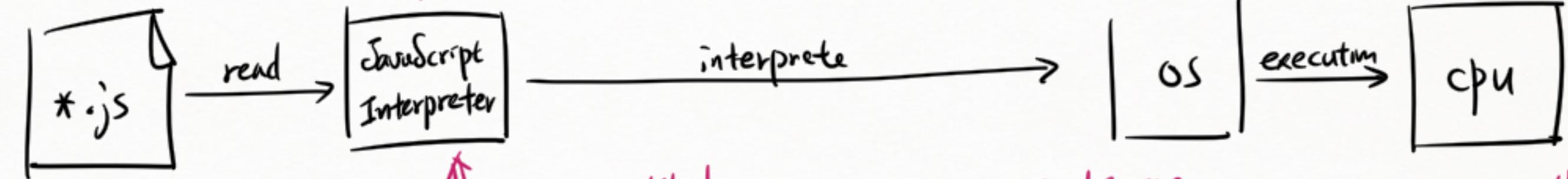
* 컴파일 방식과 인터프리트 방식

→ 컴파일 방식 :



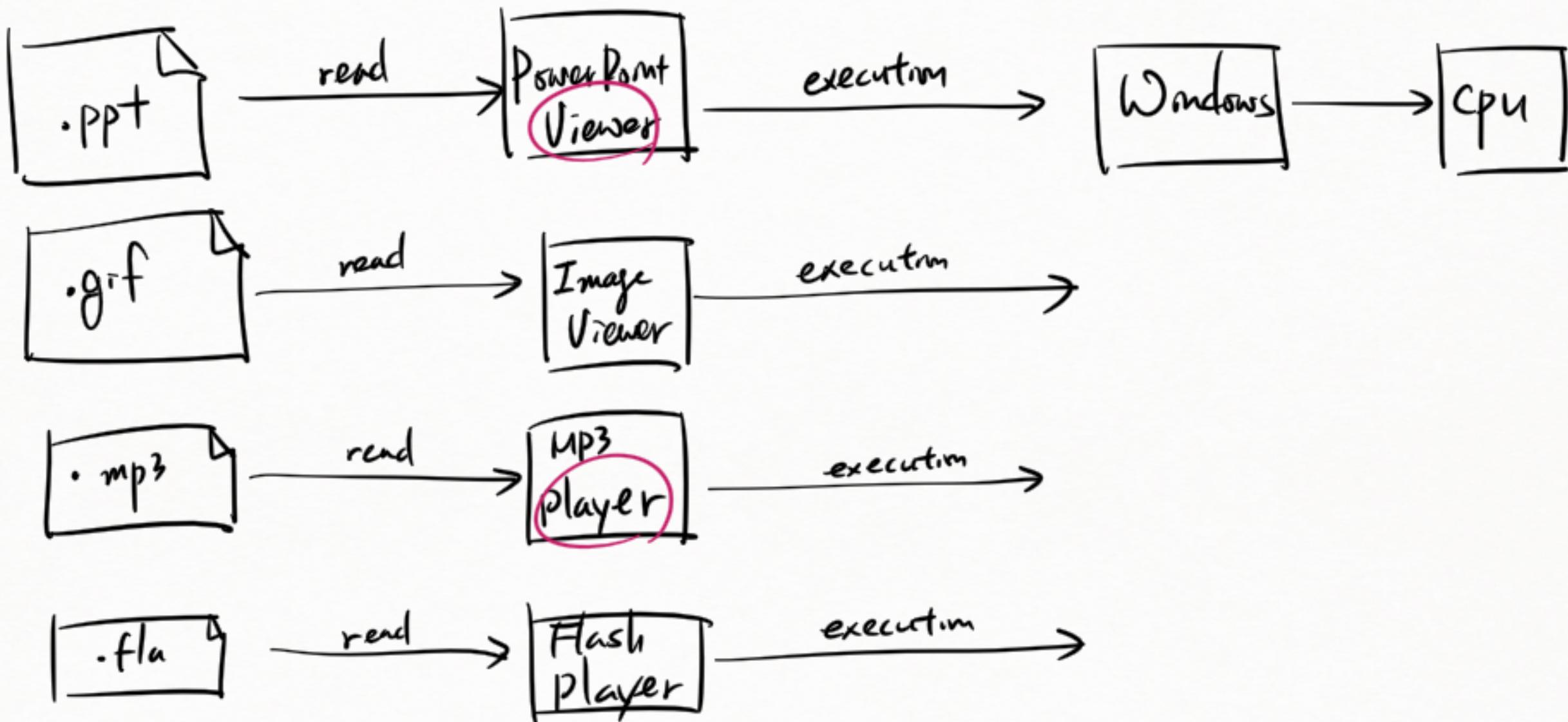
- 실행할 때 기계어 파일이 있으어야 한다.
 - 소스파일 필요없다 → 소스파일을 보호(자산)
 - 실행할 때 컴파일러 불필요.
- 기계어 바로 실행
보통 속도 빠르다

인터프리트 방식 :

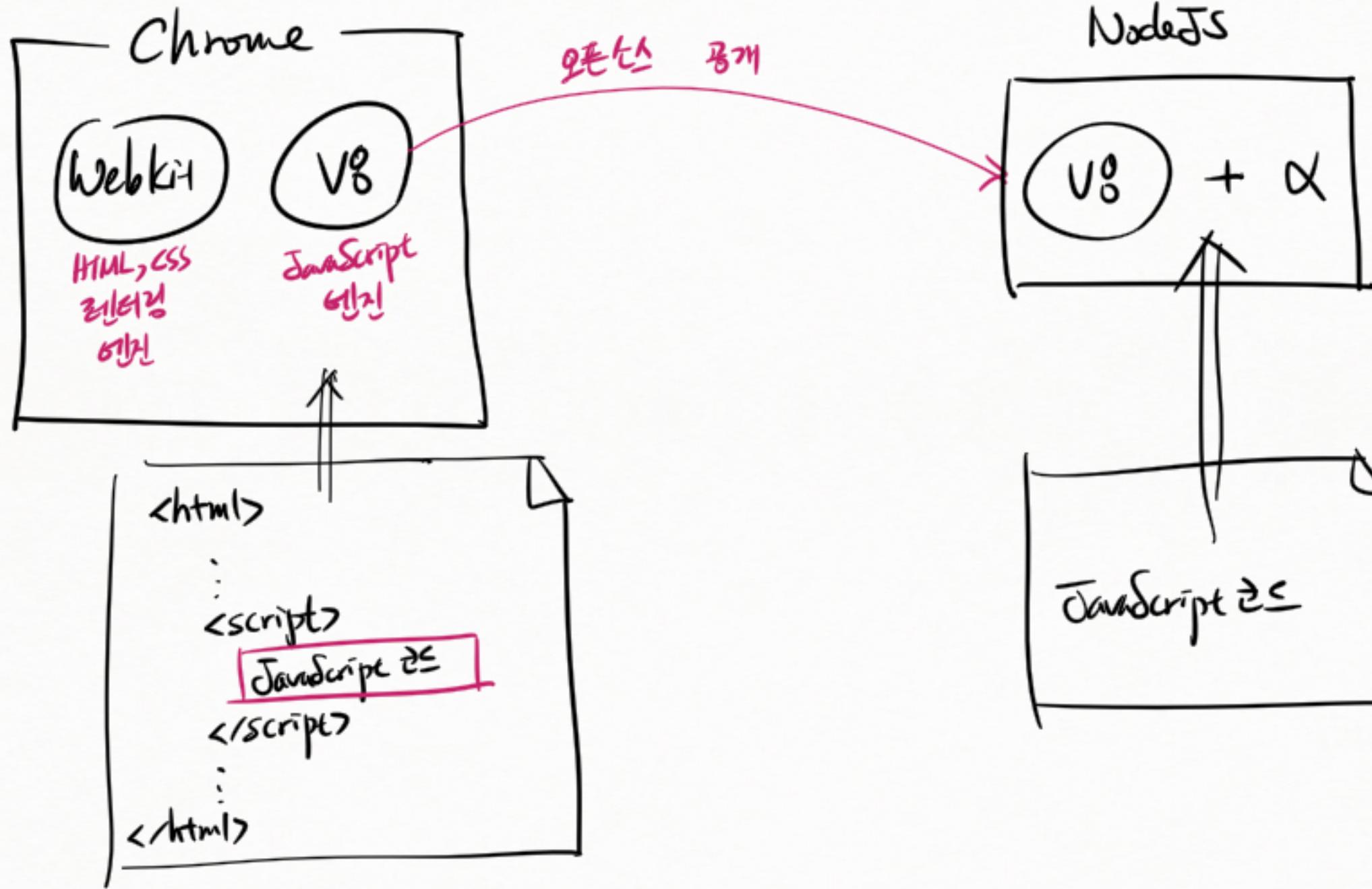


- ↑
컴파일 하지 않는다.
- 실행할 때마다 소스파일 필요
 ↓
 소스파일 보호
 ↓
 자산으로 보호하기 힘들다.
 - 실행하는데 인터프리터 필요.
- 매번 명령어를 해석하는 데
 ↓
 보통 속도 느린다.

* ~~viewer=player~~ = viewer = player = engine = virtual machine



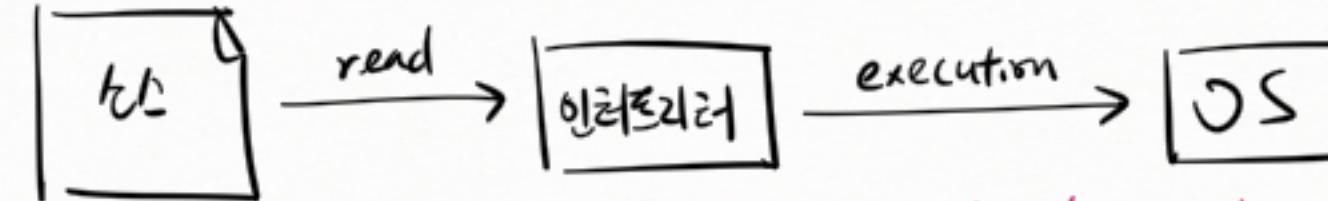
* JavaScript (Interpreter) Engine



모든스 공개
V8 + API
Standalone
JavaScript
Engine

* JIT Compile 와 AOT Compile

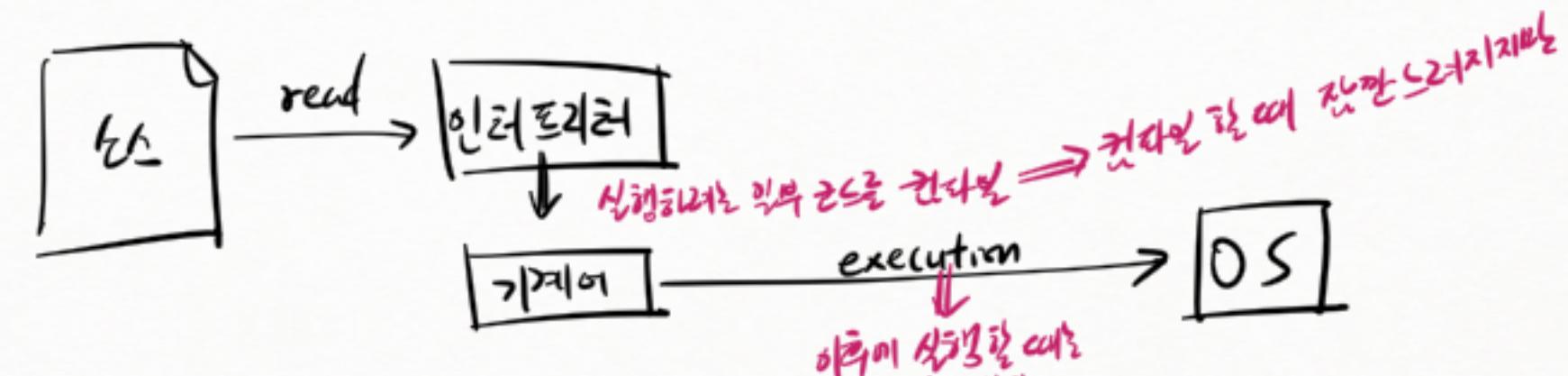
① Plain 인터프리터 :



매번 소스를 검사하고 해석하기 때문에 속도느린다

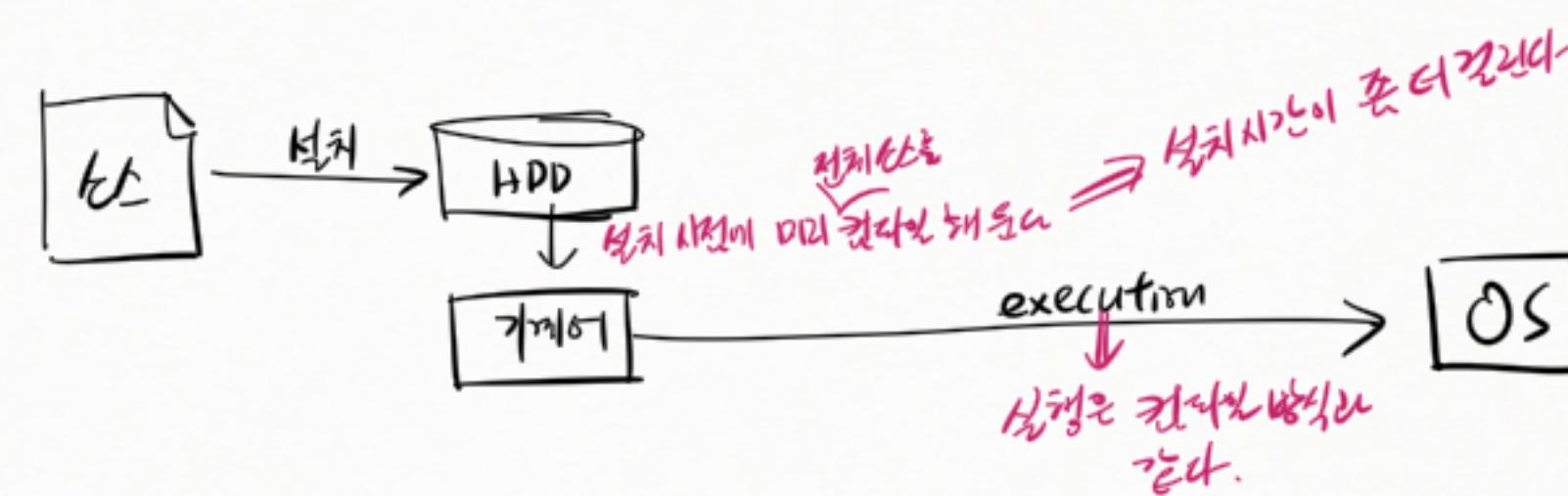
② JIT (Just in Time) 컴파일 :

바로 그 시점에
직접코드를 컴파일



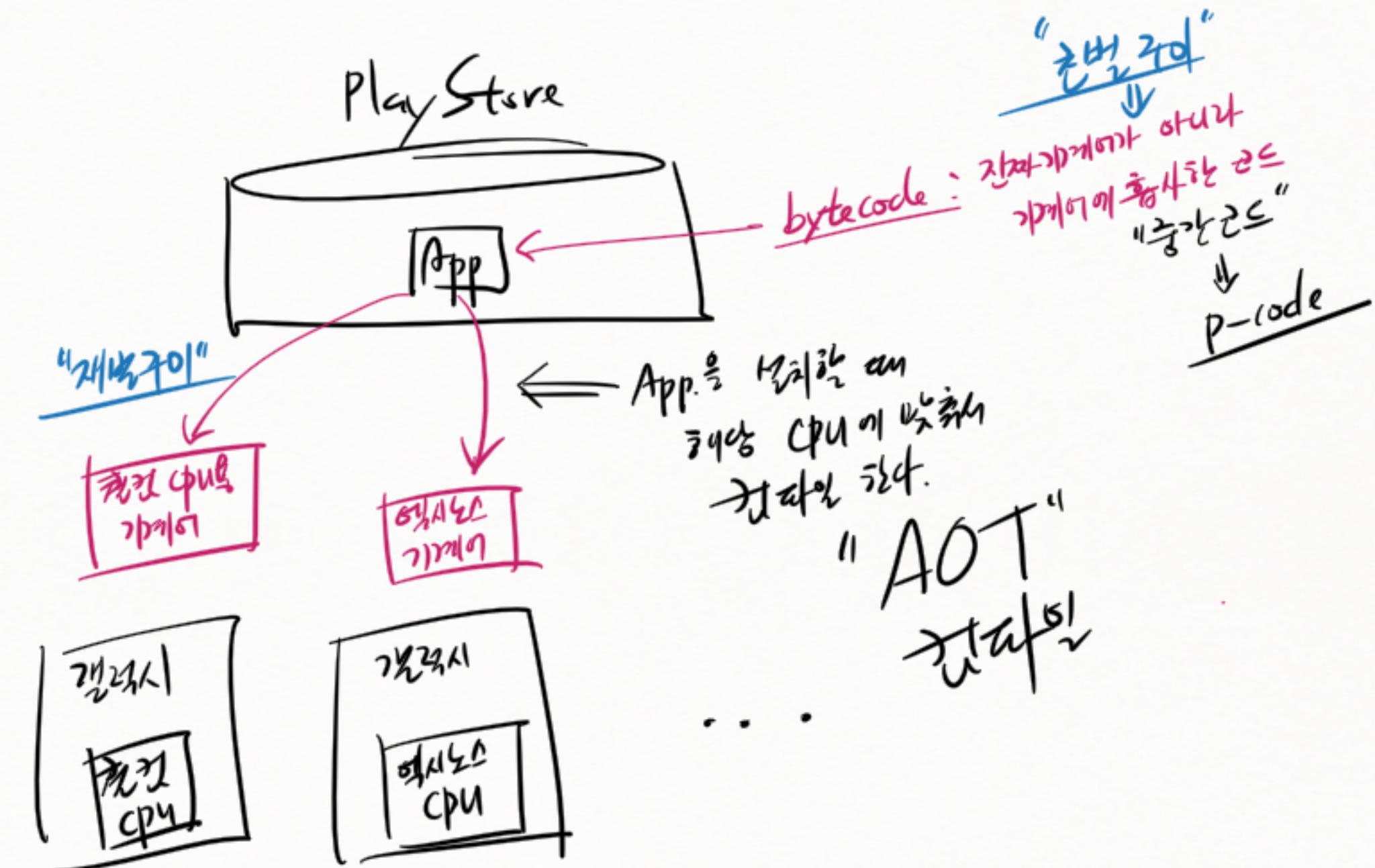
③ AOT (Ahead of Time) 컴파일 :

앞장 시점에 미리
컴파일
(예제)

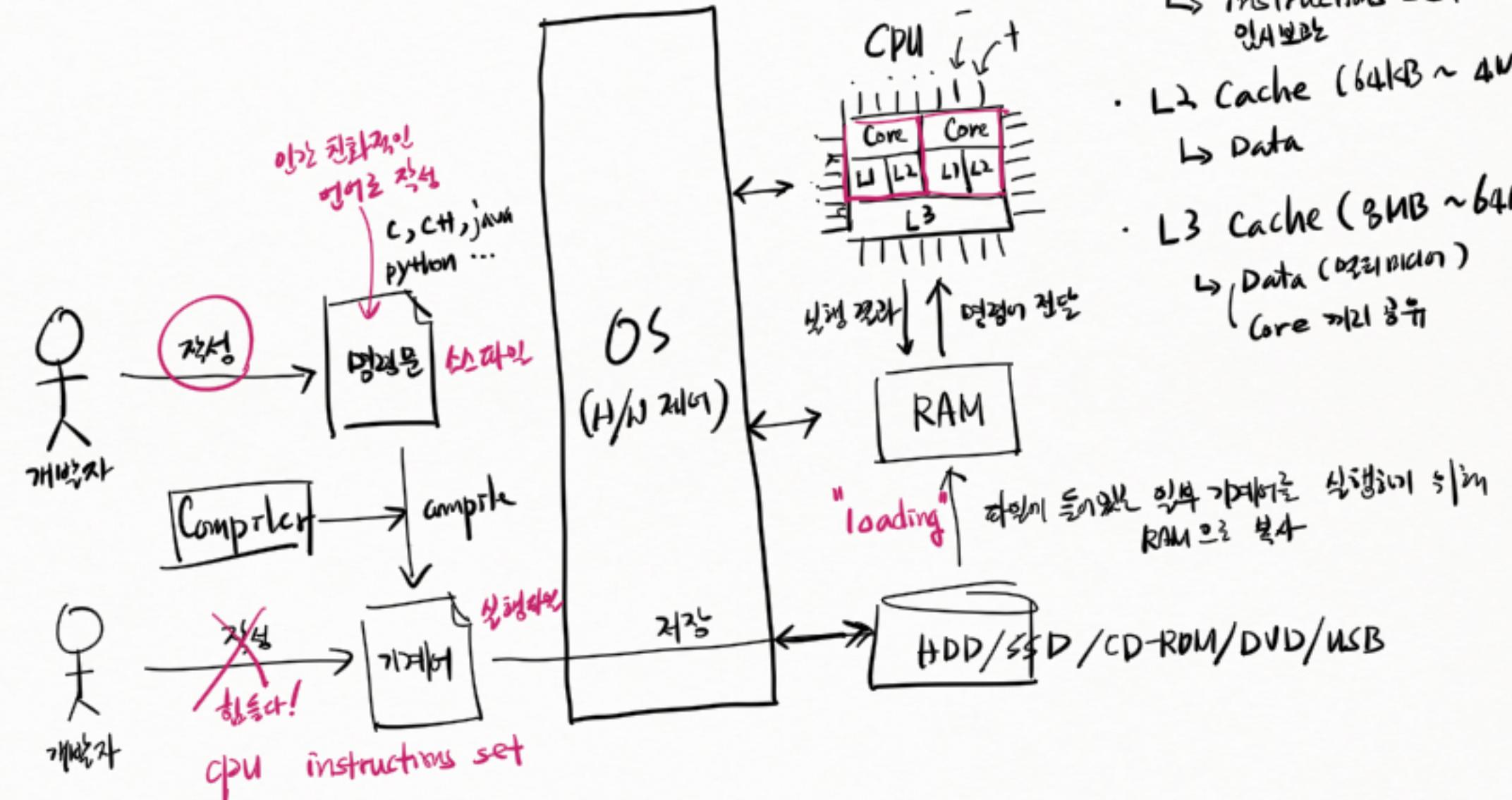


ex) Android
App

* Android 앱 App, AOT 간략화



* OS, CPU, RAM, HDD, 기계어, 명령문 간계로

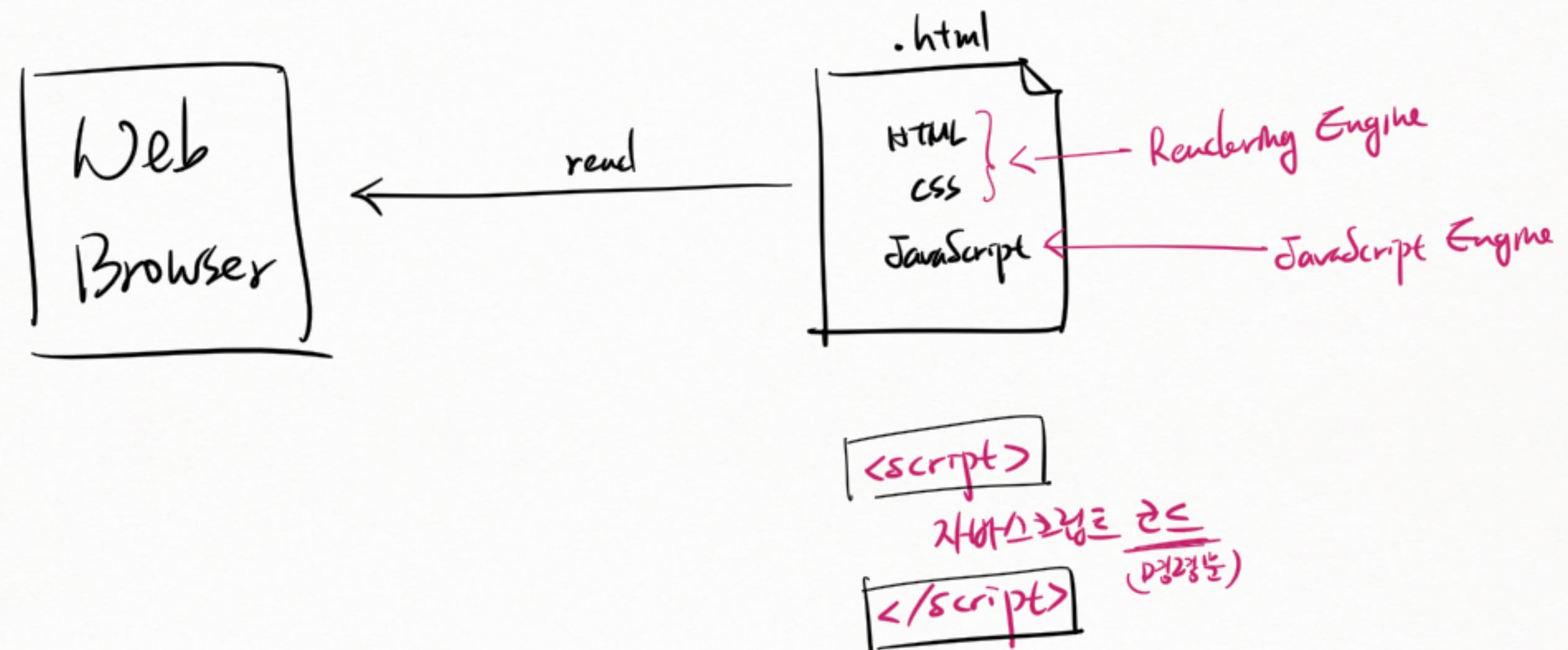


- L1 Cache (8KB ~ 64KB)
 - ↳ instructions set (명령)
 - ↳ 임시보관
- L2 Cache (64KB ~ 4MB)
 - ↳ Data
- L3 Cache (8MB ~ 64MB)
 - ↳ Data (여러 프로그램)
 - ↳ Core끼리 공유

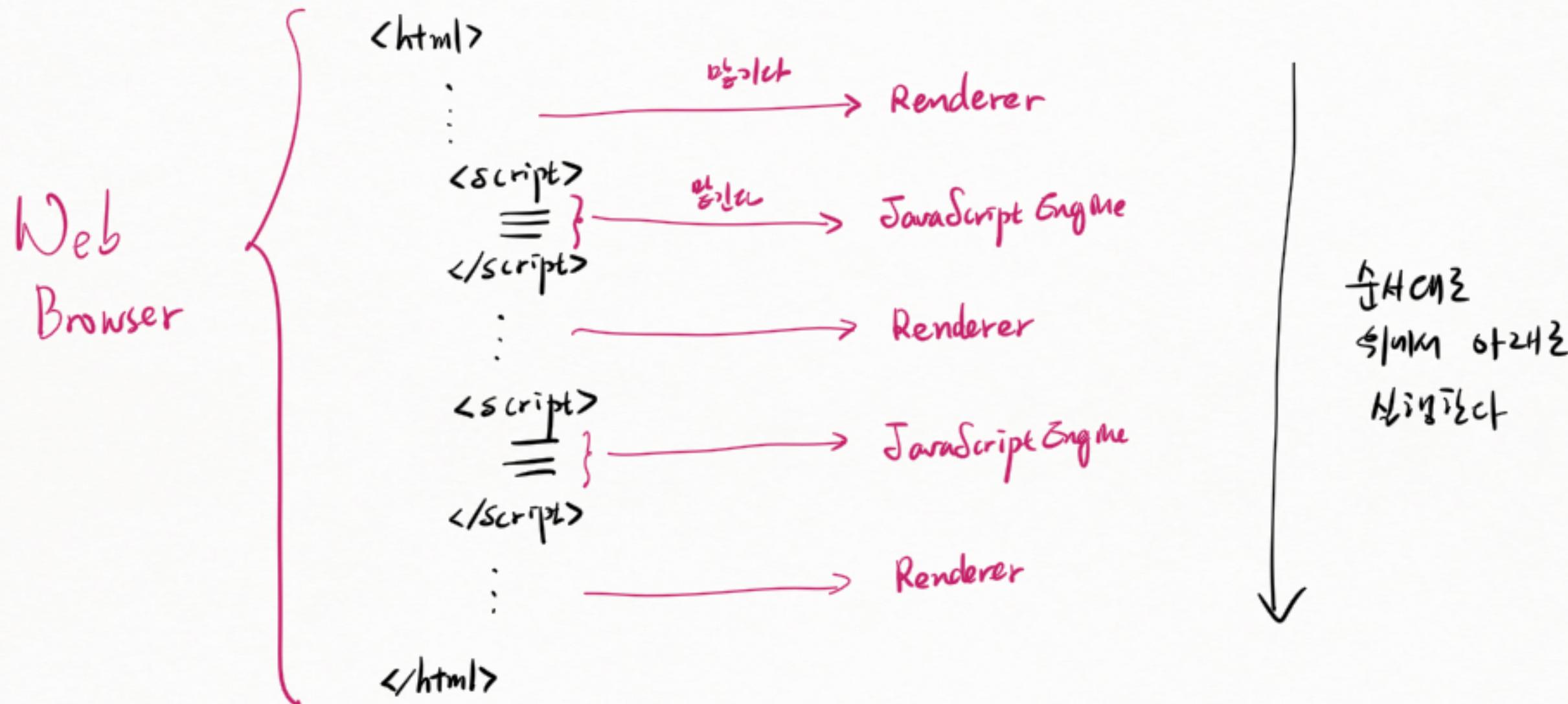
기억장치에서 일부 기계어를 실행하기 위해
RAM으로 복사

HDD/SSD/CD-ROM/DVD/USB

* Web Browser et JavaScript



* <script> 태그



JavaScript ပုဂ္ဂနယ်

① unsole ~~very~~

대한민국 서울 경기 부천 시흥 전인동
개인회생. 개인회생. 개인회생. 개인회생. 개인회생. 개인회생(1);

비즈캐스트 . 촬영하라 ("홍길동")

학생. 전화번호는 (?)

비즈캐프 . 췌아라 ("홍길동") . 전화번호록

한국
↓
진현개체

console.log(값1, 값2, 값3, ...);

↑
(2)

11

1
(기)

11

(기능을 수행하는데 있어 필요한 같은 전략)

argument (인자)
parameter (파라미터)

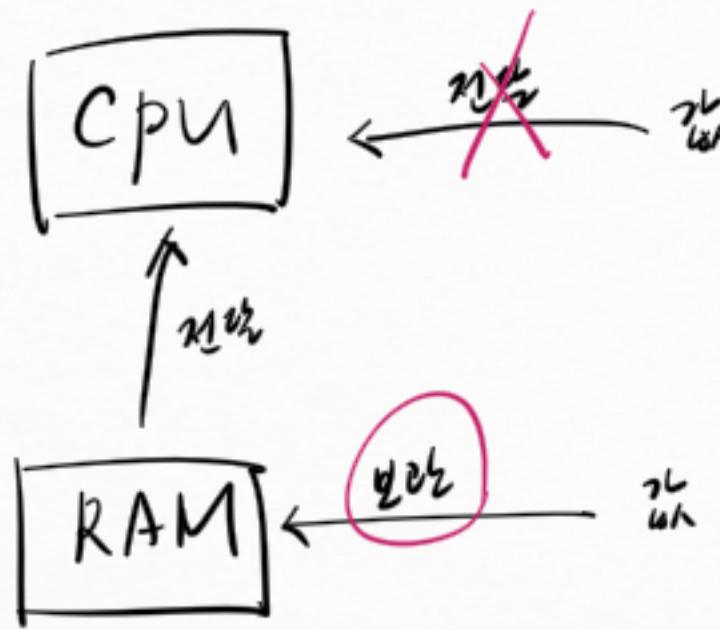
Web Browser의
인터넷 접속
설정

"aaa", } (24%) strong
"aaa", }

3.14 } (24) number

true } (y2) boolean
false

② 변수와 값



값은 저장된
변수를 표현하는
방법이다 //

"변수의 값을
저장하는
방법이다."

var 변수명;
let 변수명;
const 변수명;

↑ 흔히

변수 선언
(variables) (declaration)

변수명 = 값;
변수명 = 변수명;
변수명 = 값(1);
변수명 = 값(2);

assignment
operator

값을 저장하는
방법이다.

* = : assignment operator
할당

size
10

message
true

PI
3.14159

localVar
100

변수 = (값, 참조, 주소);
I-value
반드시 메모리에 있어야 한다
r-value

* for 문 핵심



working

true

for (① var i = 0 ; ② i < size ; ④ i++) {

① 처음 액션문의 시작
② 조건이 참이면 블록이 실행
④ i가 범위에 들어 있는 경우
i 증가시키기.

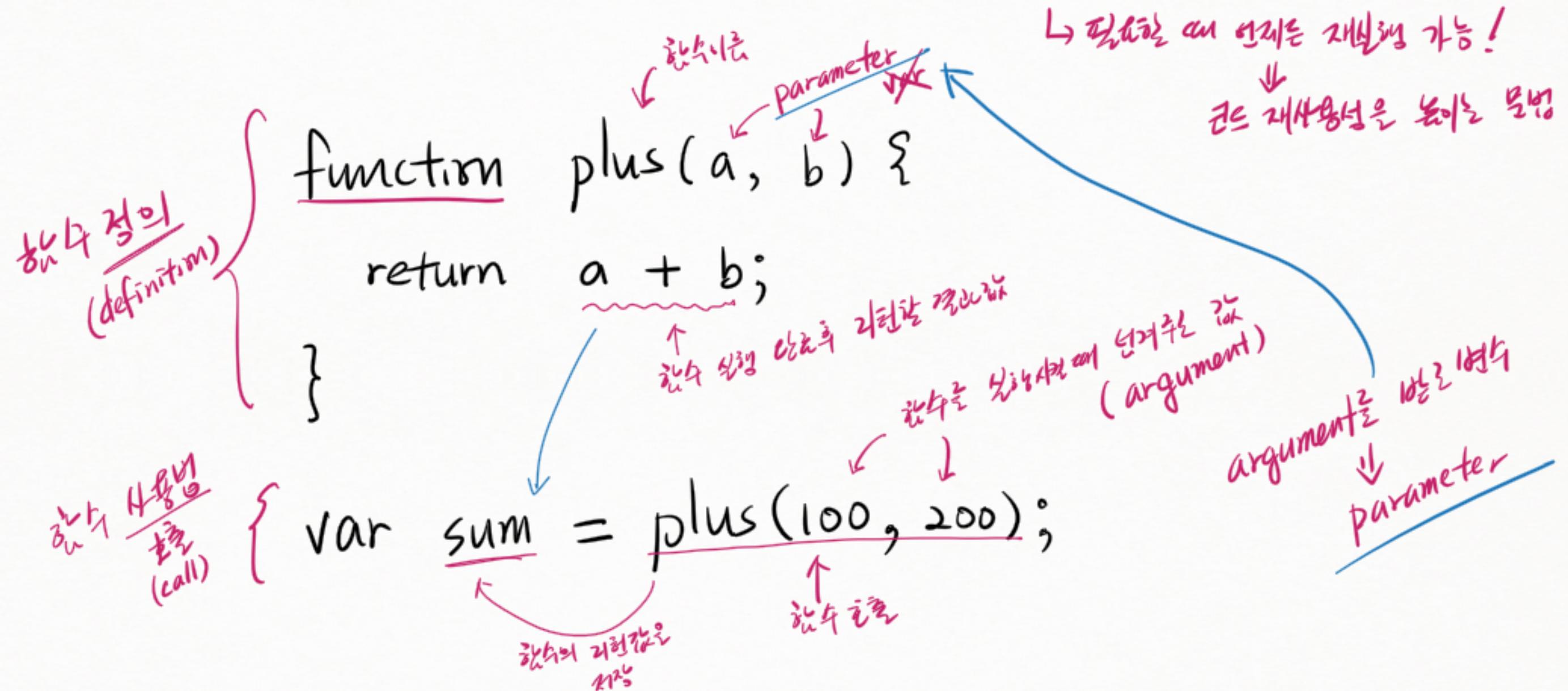
① → ② → ③ → ④
for } exit

}

비교되는 알고리즘을 통해
설명합니다. 그로
써놓은 것입니다.

* function (함수)

↳ 특정 기능을 수행하도록 짜놓은 명령을 누가나 이곳을 불인가.

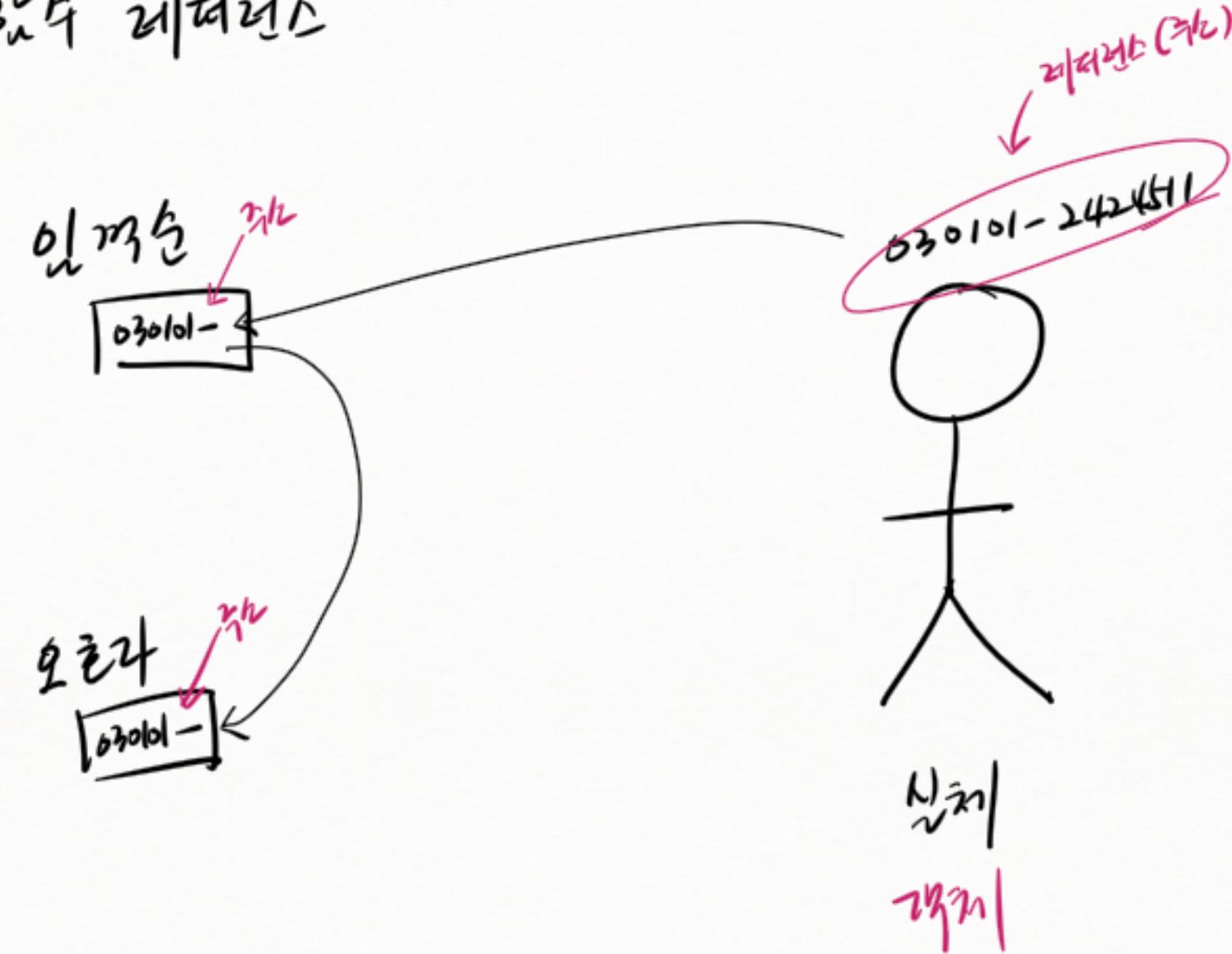


* 광범위한 예제

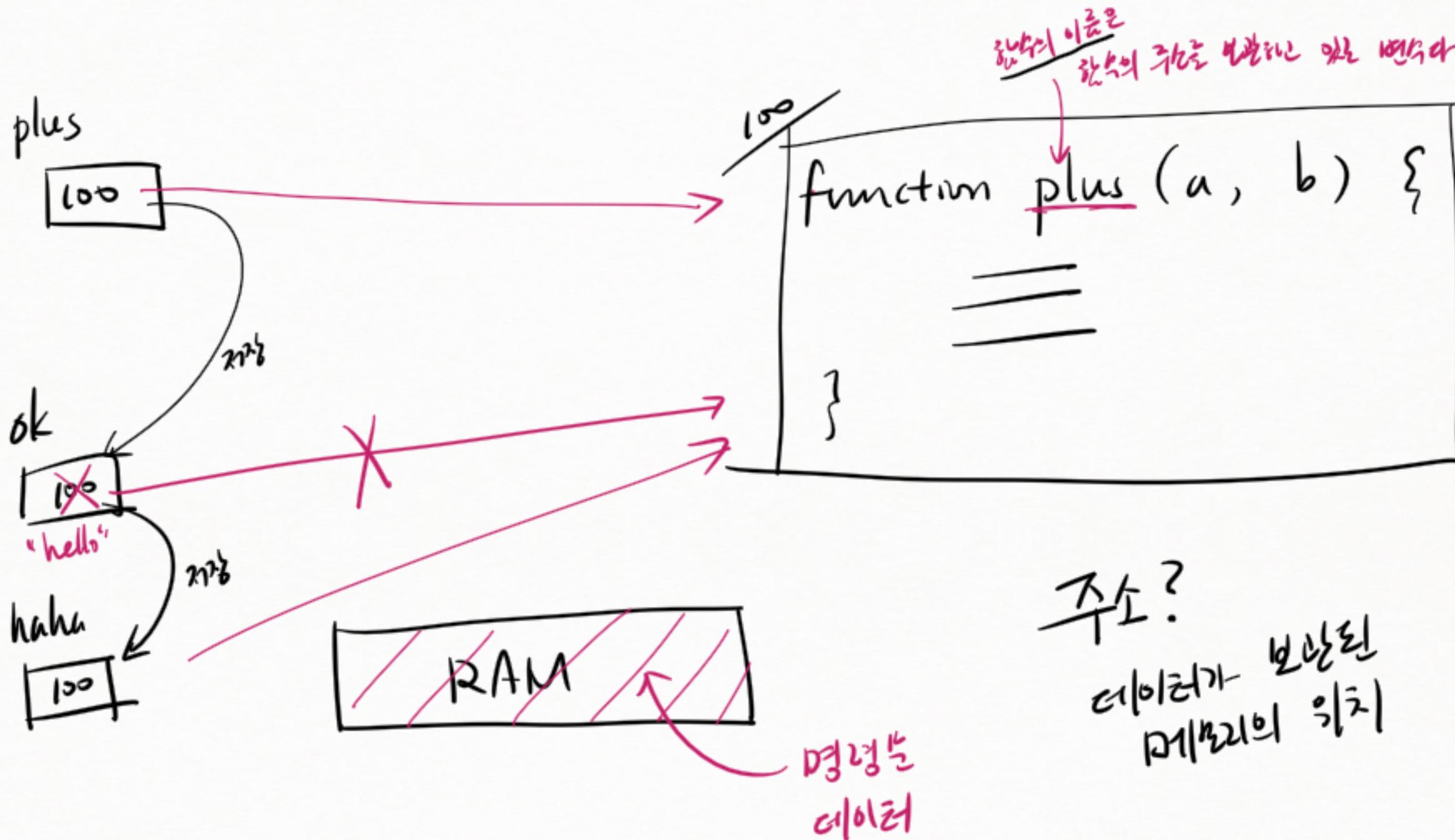
① 값: plus() 함수 이름
plus(100, 200)
↓
② 이런 값을 봉나다)

console.log(
 ③ 끝

* 키우기 퍼런스



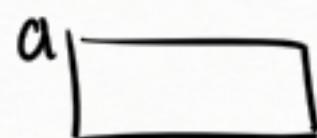
plus()
||
ok()
||
haha()



* static type binding vs dynamic type binding

Java

int a;



a = 100;

a = -20;

a ~~= "Hello";~~

기본
타입

Language 주의 프로그래밍 언어
언어 → 프로그래밍의 깊이
유지보수가 어렵다

JavaScript

var a;



a = 100; ← a는 정수 변수

a = "Hello"; ← a는 문자열 변수

a = true; ← a는 boolean 변수

유동적
변수
유형이
정해지지
않는다.

프로그래밍
언어

Script 주의 프로그래밍 언어
자유분방 → 프로그래밍의 깊이
유지보수가 어렵다

* Object : Object (변수 + 함수 + 배열)

```
var name = "홍길동";  
var age = 20;  
var working = false;
```



```
var obj = new Object();
```

obj
| 200

```
obj.name = "홍길동";  
obj.age = 20;  
obj.working = false;
```

key	value
name	"홍길동"
age	20
working	false



(Object)
인스턴스 = object

* 객체 : 데이터 (변수 + 함수 + 객체)

```
var name = "홍길동";
var age = 20;
var working = false;
```

}

→

var obj = {

name: "홍길동",
age: 20,
working: false

};

객체는
자바스크립트로
표현한 것

"object literal"

6b)

10°

key	value
name	"홍길동"
age	20
working	false

Literal (값을 변수로 표현한 것)

문자열 → "abc", 'abc'

숫자 → 10, 3.14

논리 → true, false

undefined의 값이 저장된 값이 없을 때 → undefined

null의 값이 없을 때 → null

:

* DOM API - HTML를 다루는 프로그래밍을 위한 사용하기 쉬운 인터페이스

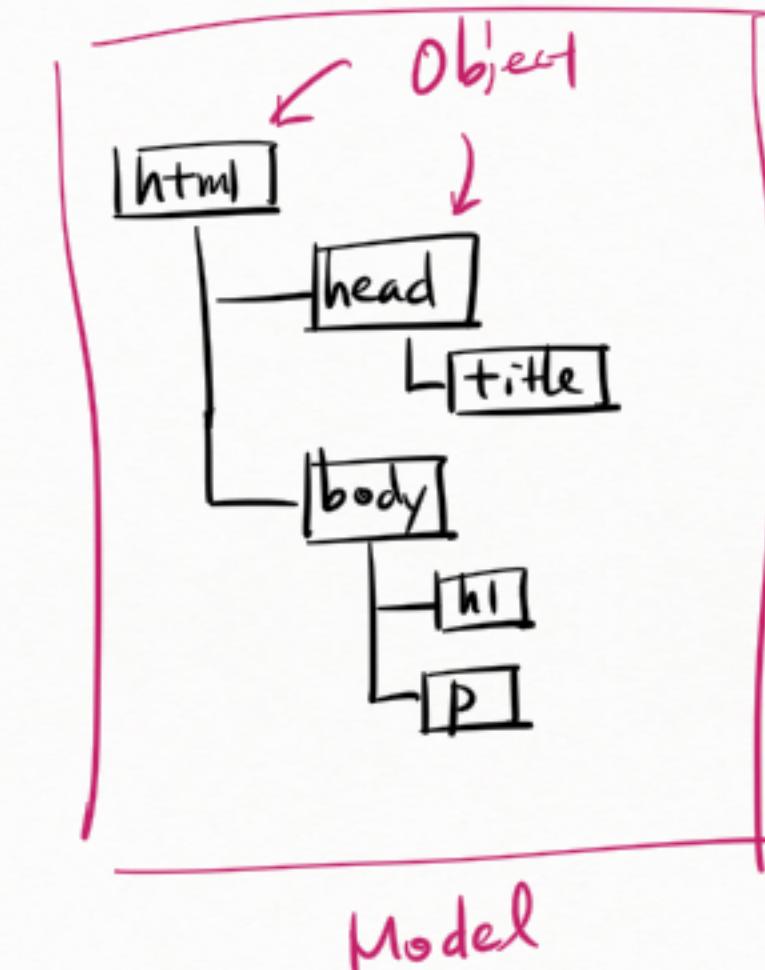
Document ← HTML

Object ← 객체

Model ← 구조화된 모델

```
<!DOCTYPE html>
<html>
  <head>
    <title> — </title>
  </head>
  <body>
    <h1> — </h1>
    <p> — </p>
  </body>
</html>
```

Document



Application Programming Interface (API)

App. API
HTML API

tag \equiv
tree 풀
구조화된 HTML
 \downarrow
tag \equiv 태그, 이름, 속성,
값으로 구성.
 \downarrow
" tag \equiv class, style
"

* 가변적
함수, 타리비러의 관계

"
bulk

"
operator

함수가 기본데이터를 다른 때
내용을 갖다

document.getElementById("태그아이디")

가변적

함수(실제 일하는 데)

함수가 일할 때 필요한 데이터
기본데이터

작업의 내용으로 추가 데이터
)

정적 명령. find("종결용")

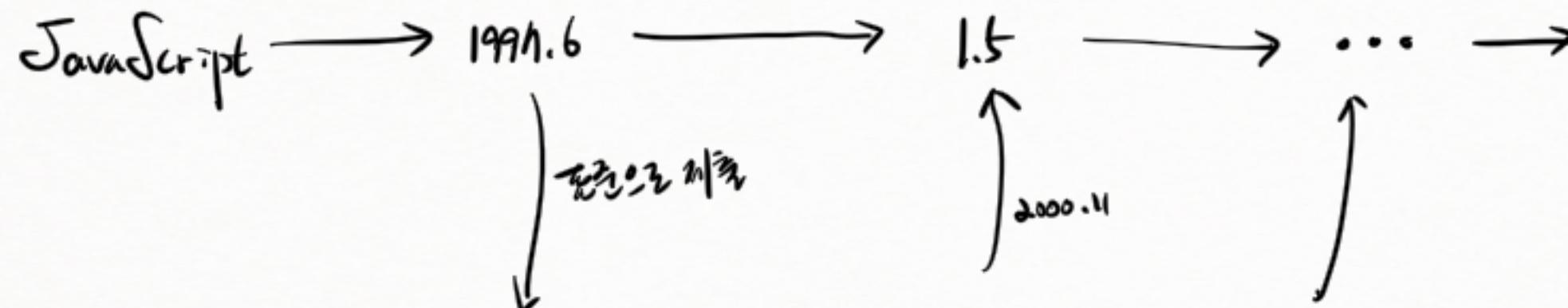
↑
기본데이터

↓

t = 작업자(operator)

정적 명령 영어화(가변적)

* JavaScript 와 ECMAScript 표준



⇒ ECMA 표준 : ECMA-262 → ECMAScript → 5.1 → 2015 → 2016 → 2017 → 2019 → ...
제3회
3rd (2011) (6) (7) (8) (9)

Adobe Flash : ActionScript → ... → X

* 태그 출기 → 이벤트 리스너 등록

The diagram illustrates the creation of an event listener on a DOM element. It starts with a variable declaration:

```
var tag1 = document.getElementById("btn1")
```

A red annotation above the code states: "document 가 가지고 있는 DOM Tree의 루트를 찾았을 때, HTML 문서에 'btn1'이라는 태그를 찾고 그 태그의 주소를 리턴". A red arrow points from this annotation to the `document.getElementById` call.

Below the variable declaration, there is a tree diagram showing the DOM structure:

```
html
└── head
└── body
```

A red annotation next to the tree says: "DOM Tree를 가리킨다". A red arrow points from this annotation to the tree.

On the right side, the variable `tag1` is assigned a function:

```
tag1.onclick = function() {};
```

A red annotation next to this assignment says: "tag1은 마우스로 클릭되었을 때, onclick 이벤트는 이벤트로 저장되는 순간에 실행된다. 만약 이 이벤트로 저장되는 순간에 아예 함수를 넣지 않으면". A red arrow points from this annotation to the assignment.

Another red annotation below the function assignment says: "'onclick' 이벤트로 초기화되는 순간의 메모리에 보관". A red arrow points from this annotation to the function body.

At the bottom, a red annotation says: "click 이벤트가 오면 리스너를 등록하는 코드". An upward arrow points from this annotation to the `onclick` assignment.

* tag의 content 키워드

```
document.getElementById('p1');
```

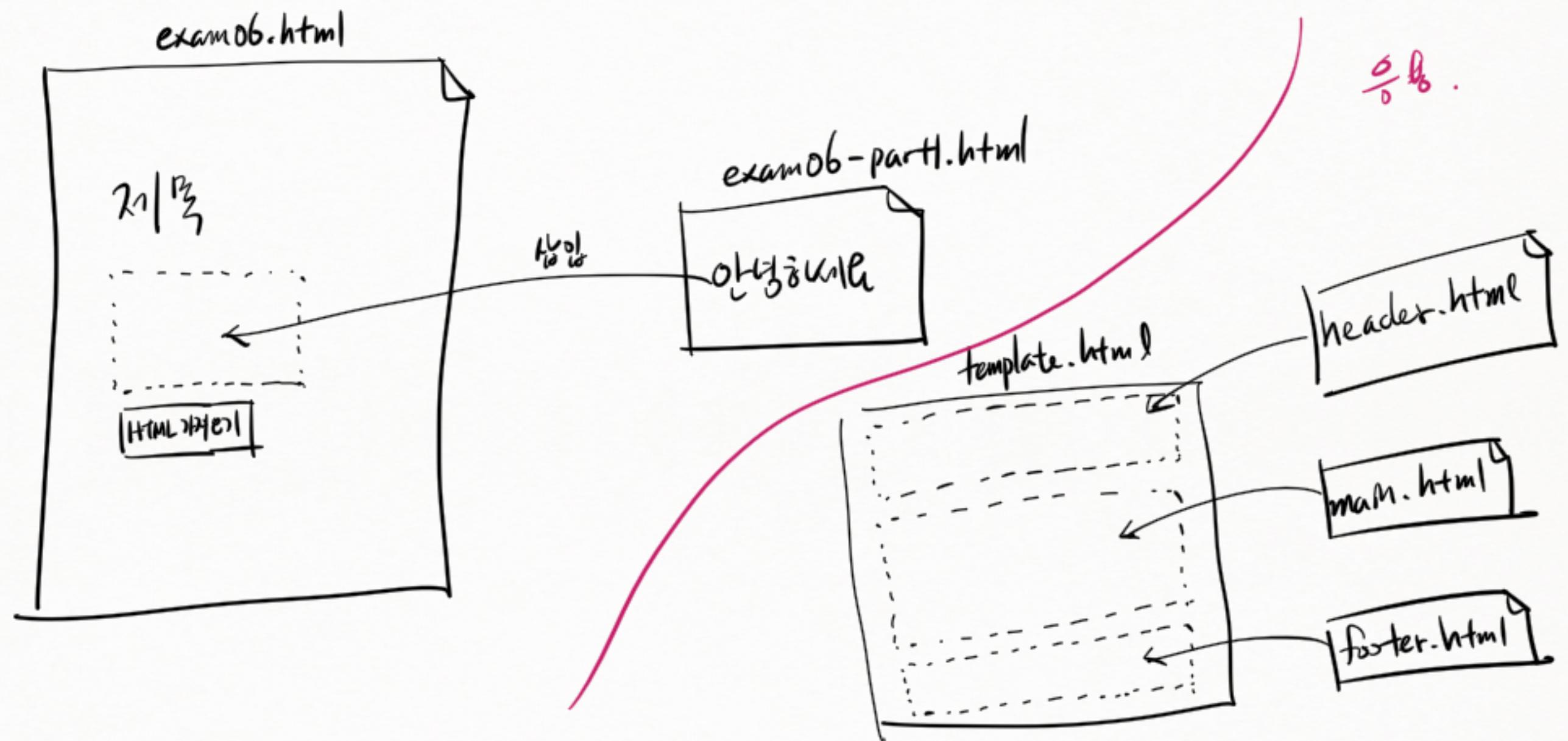


```
tag2. innerHTML = "안녕하세요";
```

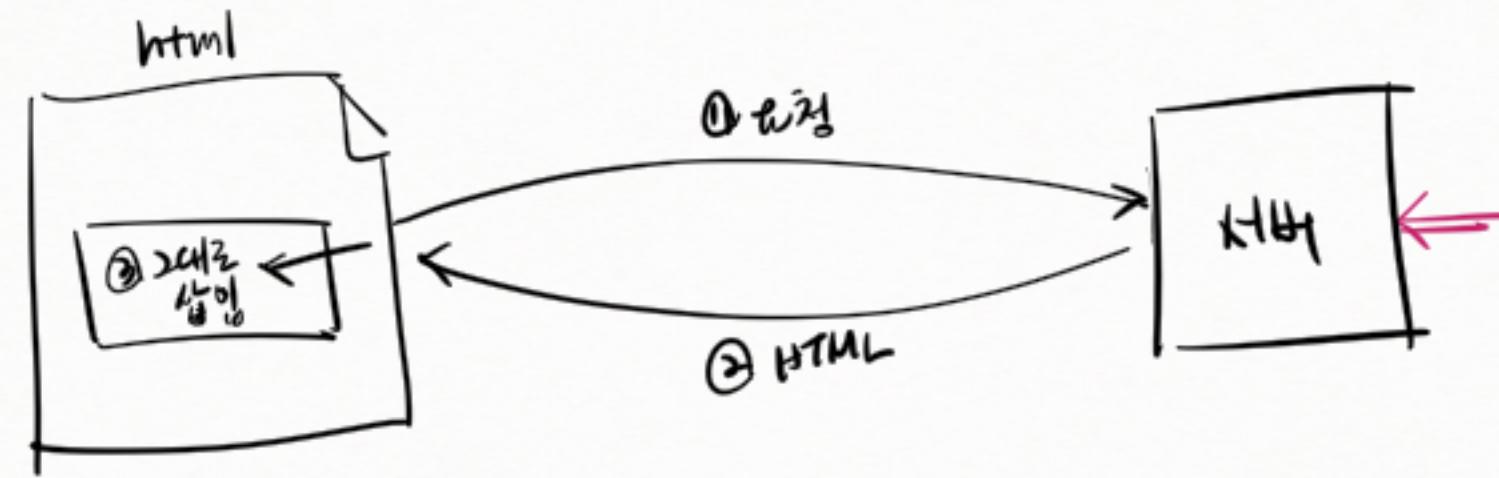
```
<p id='p1'> 하하하 </p>
```



* 파일 티플릿 구조

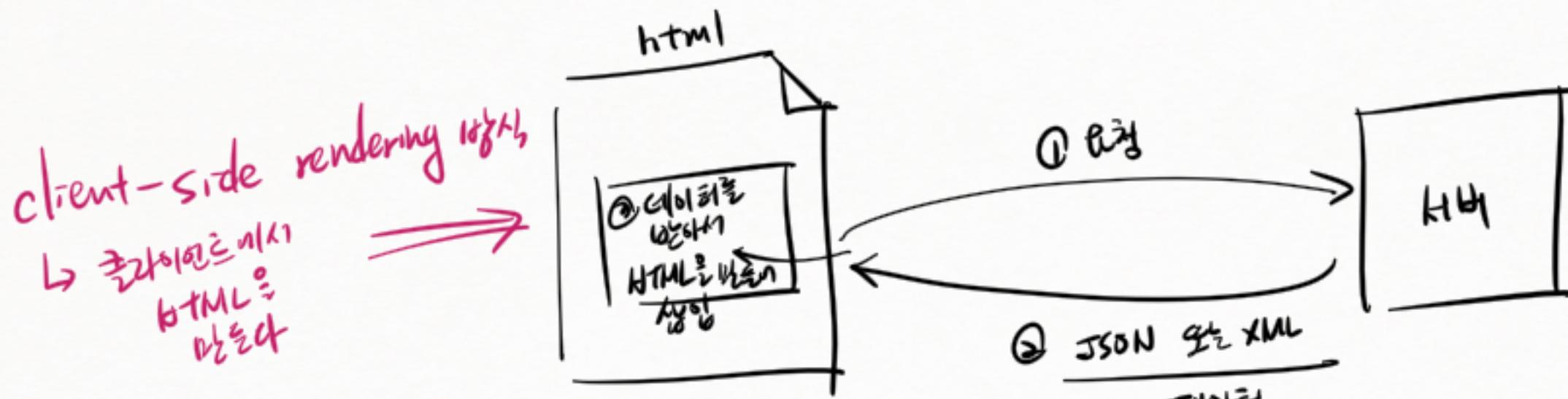


* 사이버 차원을 넘겨 : "server-side rendering" vs "client-side rendering"



Server-side rendering 방식
↳ 사이버에서 HTML을 만들어 준다

예) JSP, Thymeleaf Ⓜ



client-side rendering 방식
↳ 클라이언트에서 HTML을 만들다

예) AJAX 기법
↳ React, Angular Ⓜ

* JSON 토익

```
var obj = {  
    name: '홍길동',  
    age: 20,  
    working: false,  
    hell: function() {} —}  
};
```

object literal

JavaScript의
object literal 구조를
토익에서 활용할 때

{
 "name": "홍길동",
 "age": 20,
 "working": false
}

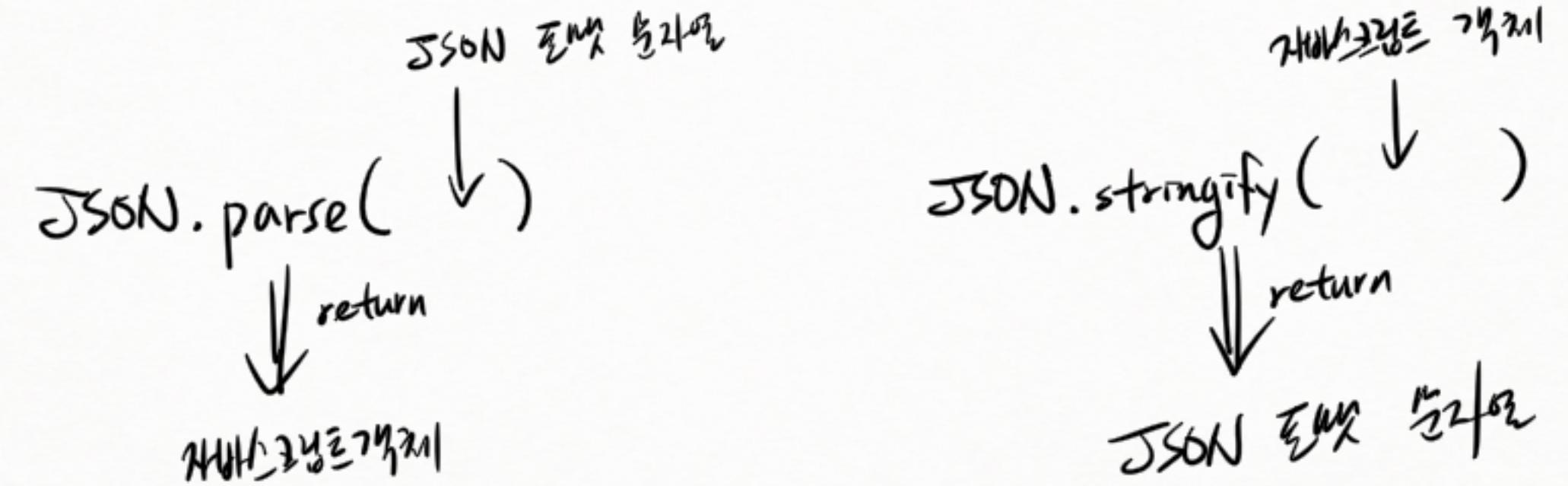
기존 (obj) property 이름을
문자열로 표기
문자열은 "" 으로 표현

문자열은 표현할 수 있다
변수로 가능

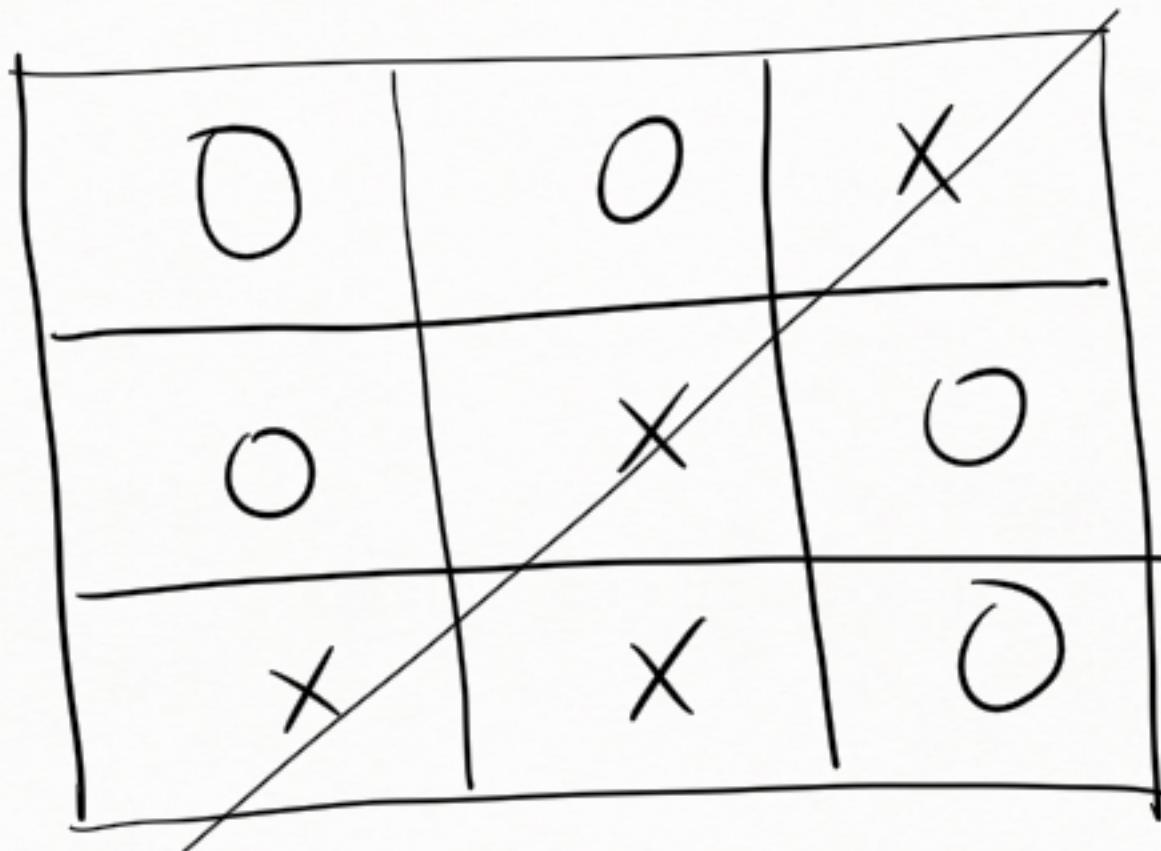
JavaScript
Object
Notation

"JSON"

* JSON Built-in API



* 2121 : tictactoe 게임 만들기
(2022-12-02)

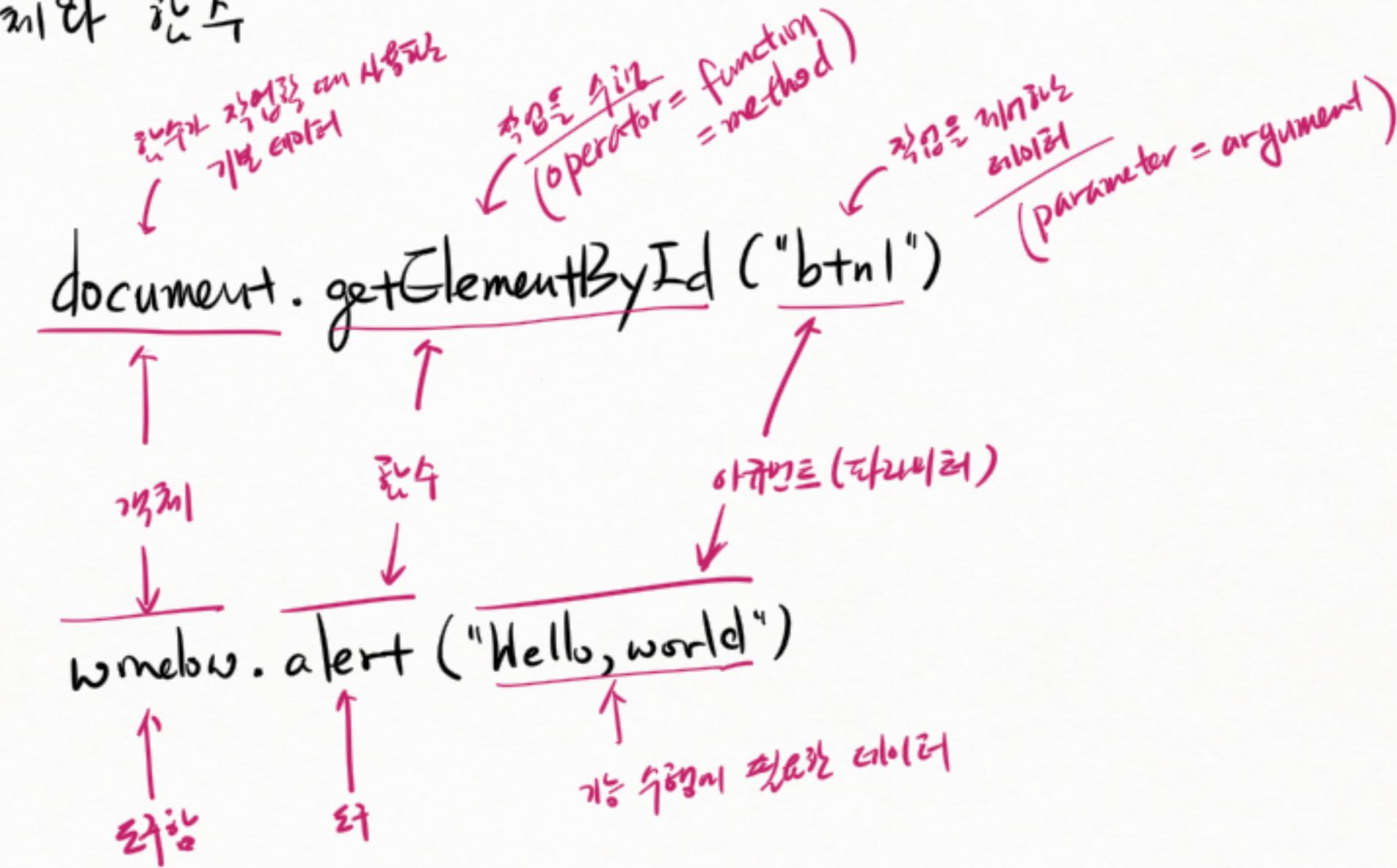


X가 이겼습니다!

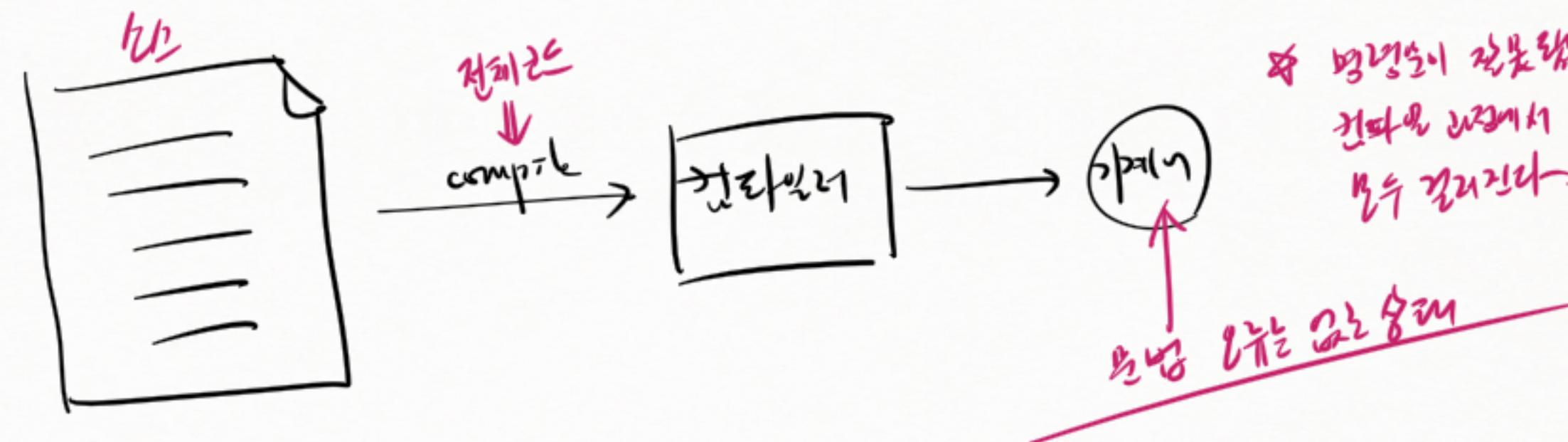
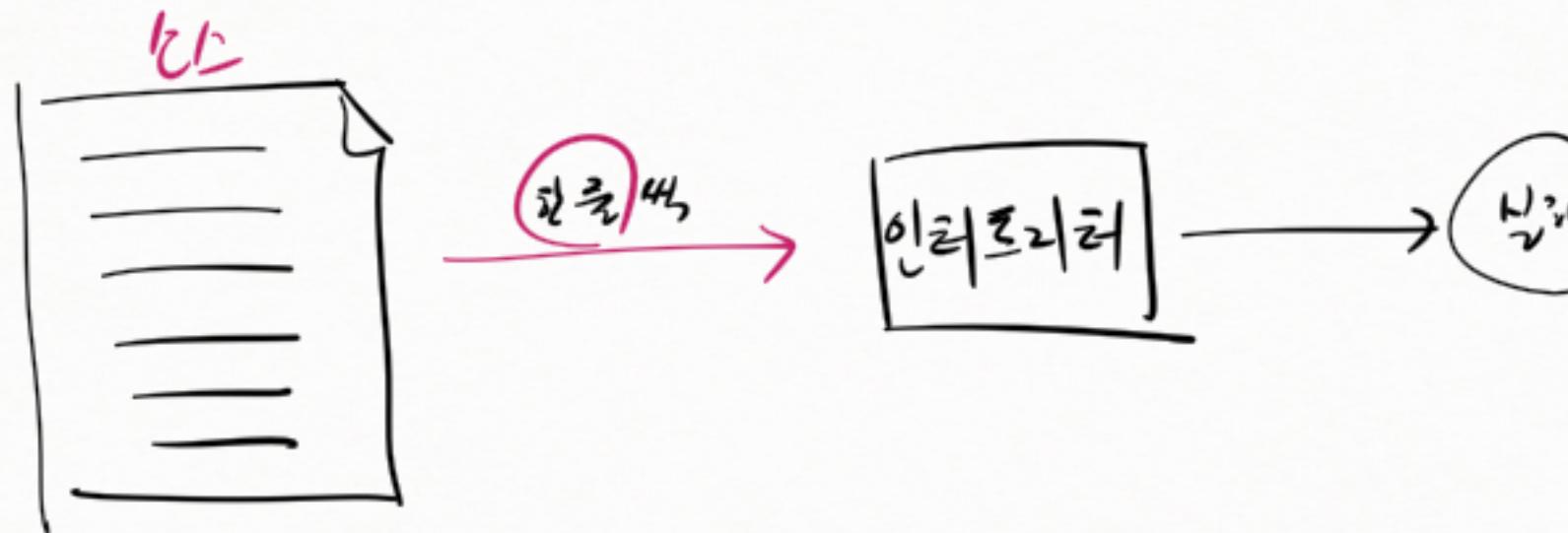
O이 이겼습니다!

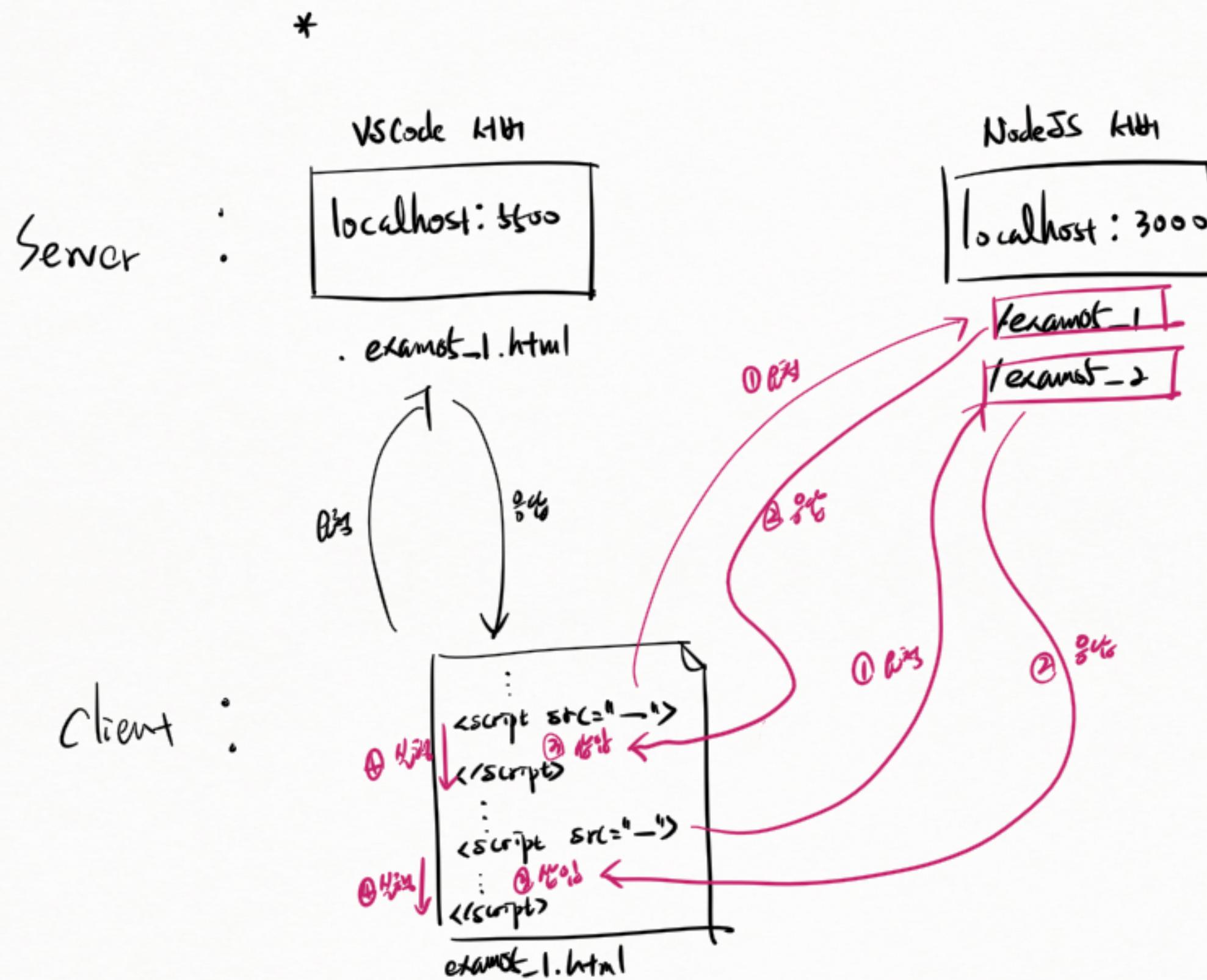
무승부입니다!

* 구조화된 문장



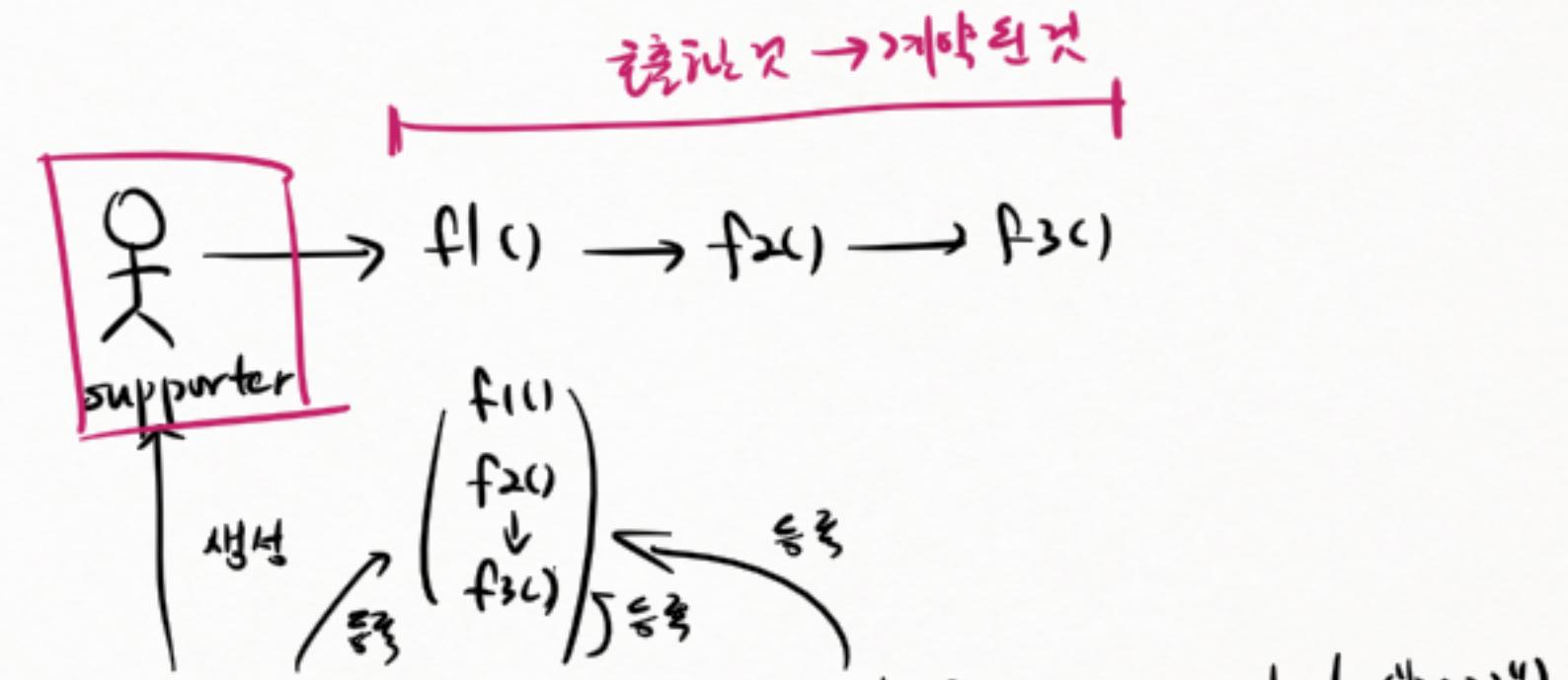
* 인터프리트 방식과 컴파일 방식





* Promise

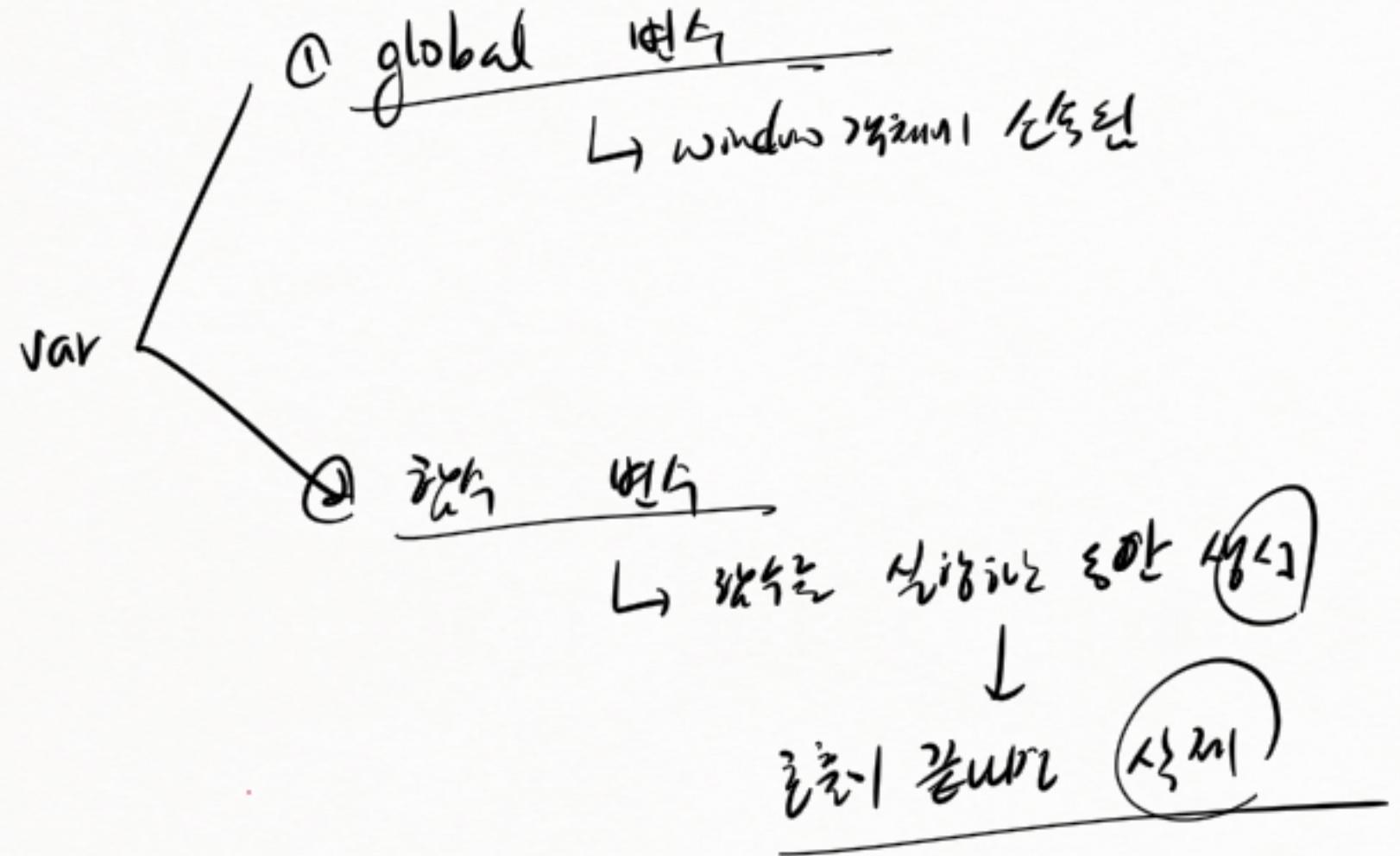
```
console.log("1111"); ← main  
var p1 = new Promise(f1);  
p1.then(f2);  
p1.finally(f3);  
console.log("2222");
```



console.log("1111") → new Promise(f1) → then(f2) → finally(f3) → console.log("2222")

Asynchronous (동시진행)

* var 변수와 블록



+ let 변수 이 부록



부록에 진입 : 부록에 진입 → 변수
부록에 진입 → 변수

let a;
{
 let a = 100;
}

* const မျဉ်

~~const v1;~~

v1

မျဉ် မှုပါန + အာရုံစွဲ

const v1 = "hello";

"initializer"

မျဉ်မှုပါန အာရုံစွဲ

* 例의 알고리즘

① const



② let



③ var

const obj = new Object();

obj

200

obj["name"] = "홍길동";

obj.age = 20;

obj.working = false;

obj ≠ new Object();

key	value
name	"홍길동"
age	20
working	false
:	:

↑
기존
값이
있으므로
새로
만들어
주면
다른
값이
된다.

key	value

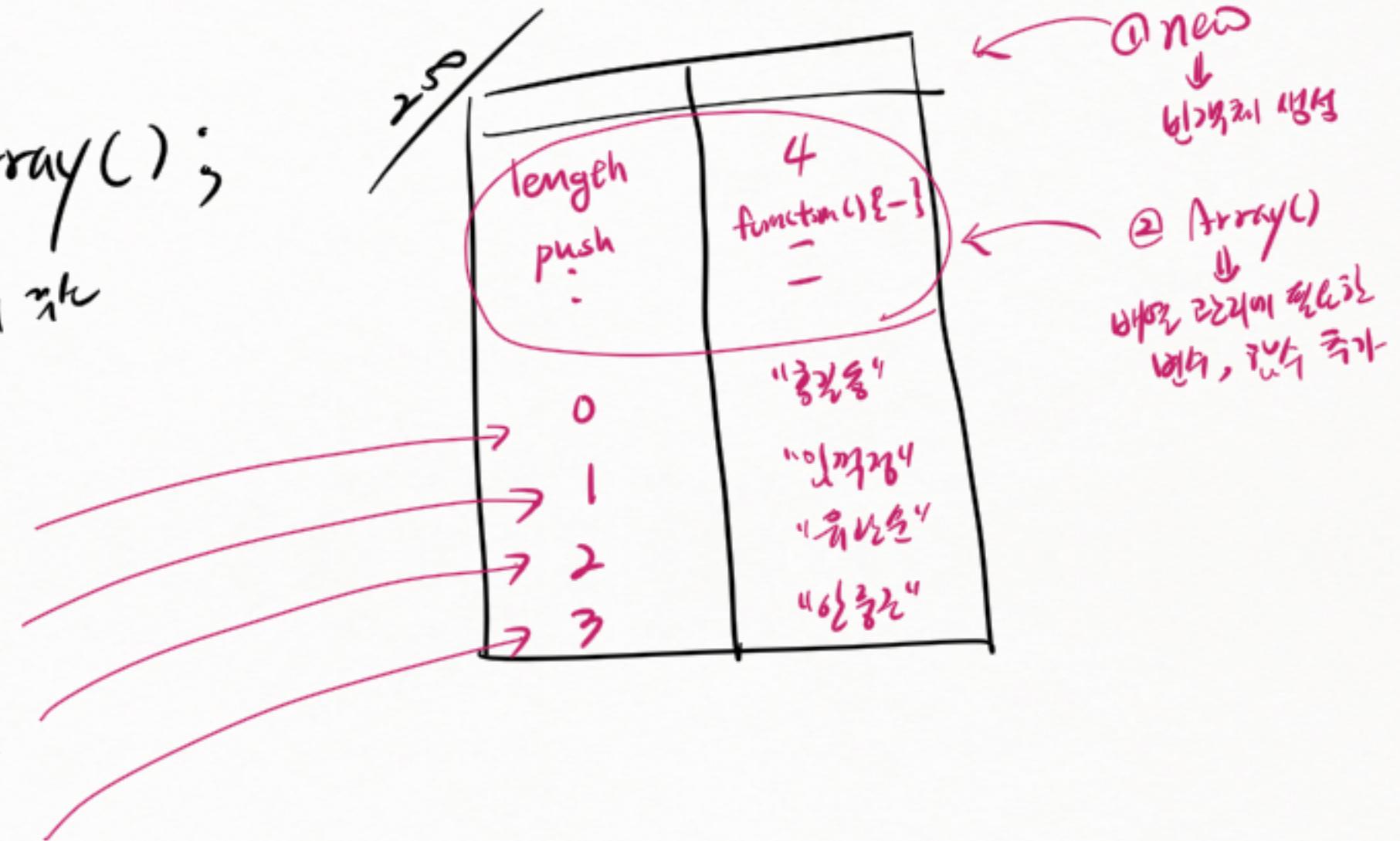
* 배열 만들기

↳ 예전 모의 변수를 선언해 위치 정하기

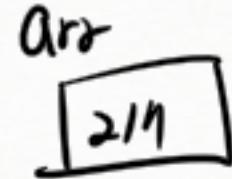
```
var arr = new Array();  
arr  
200
```

↑
변수에 할당된 주소

```
arr.push("홍길동");  
arr.push("이꺽정");  
arr.push("유비");  
arr.push("아첨자");
```



* 0.14% O in MnRu 3/2



array[38] → 211 = 211 + 0 → arr[0]

sym " $\rightarrow 218 = 219 + 1 \rightarrow$ $\alpha \vee [y]$

String " → arr[*y*] " = *y* + " → arr[*y*]

~~arr[0] = arr[1] + 1~~ → arr[1]

$$1^{224} \rightarrow 2^{24} = 2^{11} + 7 \rightarrow \text{arr}[7]$$

5月22日 11:00-12:00

~ 여기서 시작 주제를 가지는
여러가지 항목을 가리키기 때문에
이택스는 이우리 시작된다.

* 문법

operators

↳ 반복연산자: +, -, *, /, %
비교 연산자: >, >=, <, <=, ==, !=

:

기타문

Literal

→ 값은 표현하는 문법 예) "aaa", 'aaa', 10, 3.14, true/false ...

Variables

→ 값을 저장하는 메모리를 준비하는 문법 예) var/let/const/[]

{
if ~ else
switch

white
for

) 분기문 → 명령문의 실행흐름을 제어하는 문법

) 반복문

function

→ 명령을 $\left(\begin{array}{l} \text{재사용할 수 있도록} \\ \text{관리하기 편하도록} \end{array} \right)$ 가능 단위로 묶는 문법

object

→ 데이터를 다루기 편도록 그룹으로 묶는 문법

→ 함수들을 관리하기 편도록 역할 단위로 그룹화하는 문법

module

→ 객체와 함수들을 재사용하기 편하도록 그룹으로 묶는 문법

- ① $\text{temp} = y$
- ② $y = y + 1$
- ③ $y = \text{temp}$

$y = 100;$

현재 $\leftarrow y = \cancel{y++};$

$y = \cancel{100};$

① r-value 를 찾는 방법
② = 현재 값

$\underline{y = y + 1; \quad 101}$



~~$y++$~~

① 현재 y 값을
이전에 옮기다

100

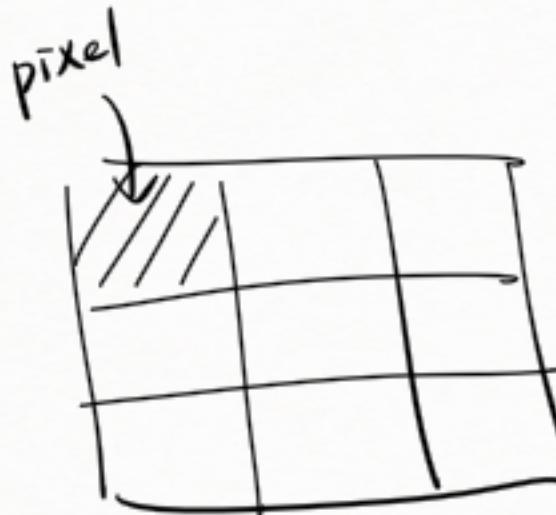
↓

② y의 값을
증가시킨다

$y \boxed{101}$

* 0(회자)와 1(이동) 수

~~255 255 255
254 255 251~~



천만화소 \div 28MB

→ 8비트 이미지 1Byte

압축 encoding

JPEG

TIFF

비트맵

decompress

비트맵

압축

① 흑백 사진 \Rightarrow 1 픽셀 크기: (0: 검은색, 1: 회색) 1bit

\Rightarrow 9 픽셀 크기: 1 픽셀 \times 1bit

$$9 \text{ 픽셀} \times 1 \text{ bit} = 9 \text{ bit}$$

② 컬러 사진 \Rightarrow 1 픽셀 크기: RGB 3 byte
(원본사진)

$$9 \text{ 픽셀 크기}: 9 \times 3 = 27 \text{ byte}$$

③ HD급 크기 사진 = $\frac{(1920 \times 1080) \times 3}{2,013,600 \text{ 픽셀}} = 6,220,800 \text{ byte}$
 $= 6,045 \text{ KB}$
 $= 5.93 \text{ MB}$

$$\begin{aligned}\text{천만화소급 사진} &= 10,000,000 \times 3 \\ &= 30,000,000 \\ &= 28 \text{ MB}\end{aligned}$$

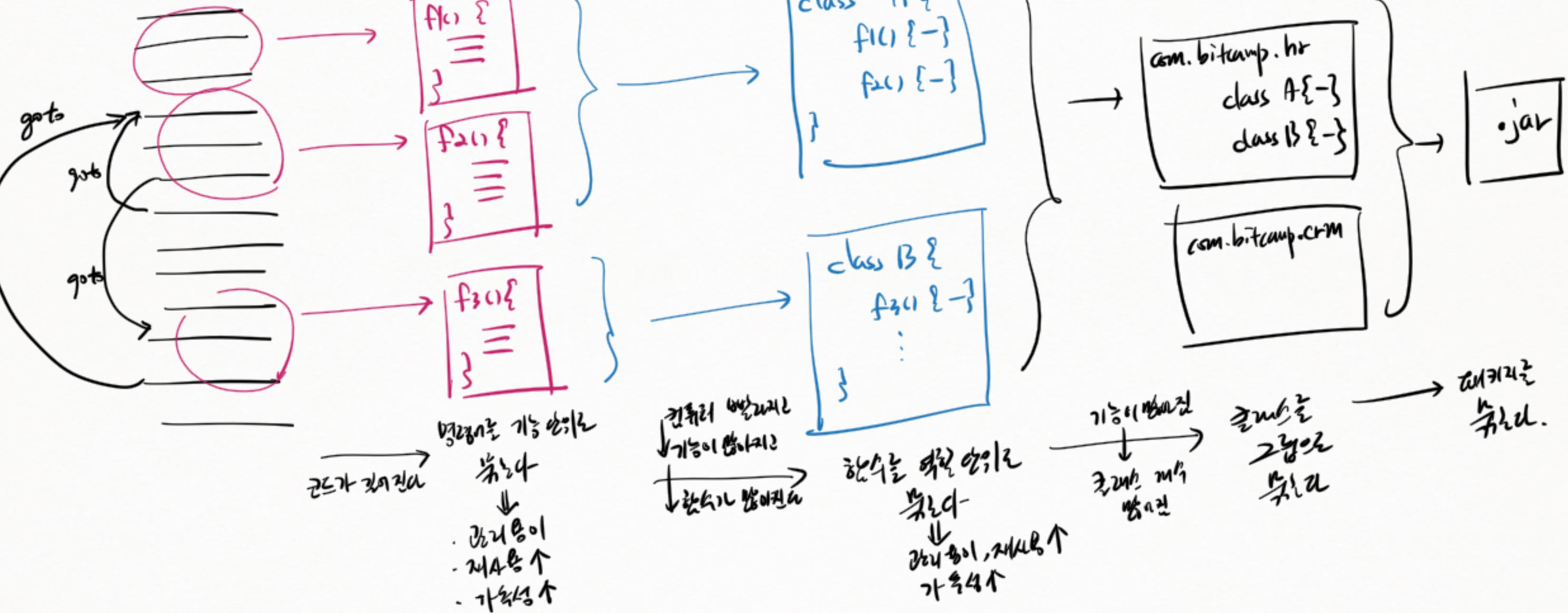
* 프로그램의 트렌드

① 짧자작 프로그램 → ② function → ③ class → ④ package → ⑤ module

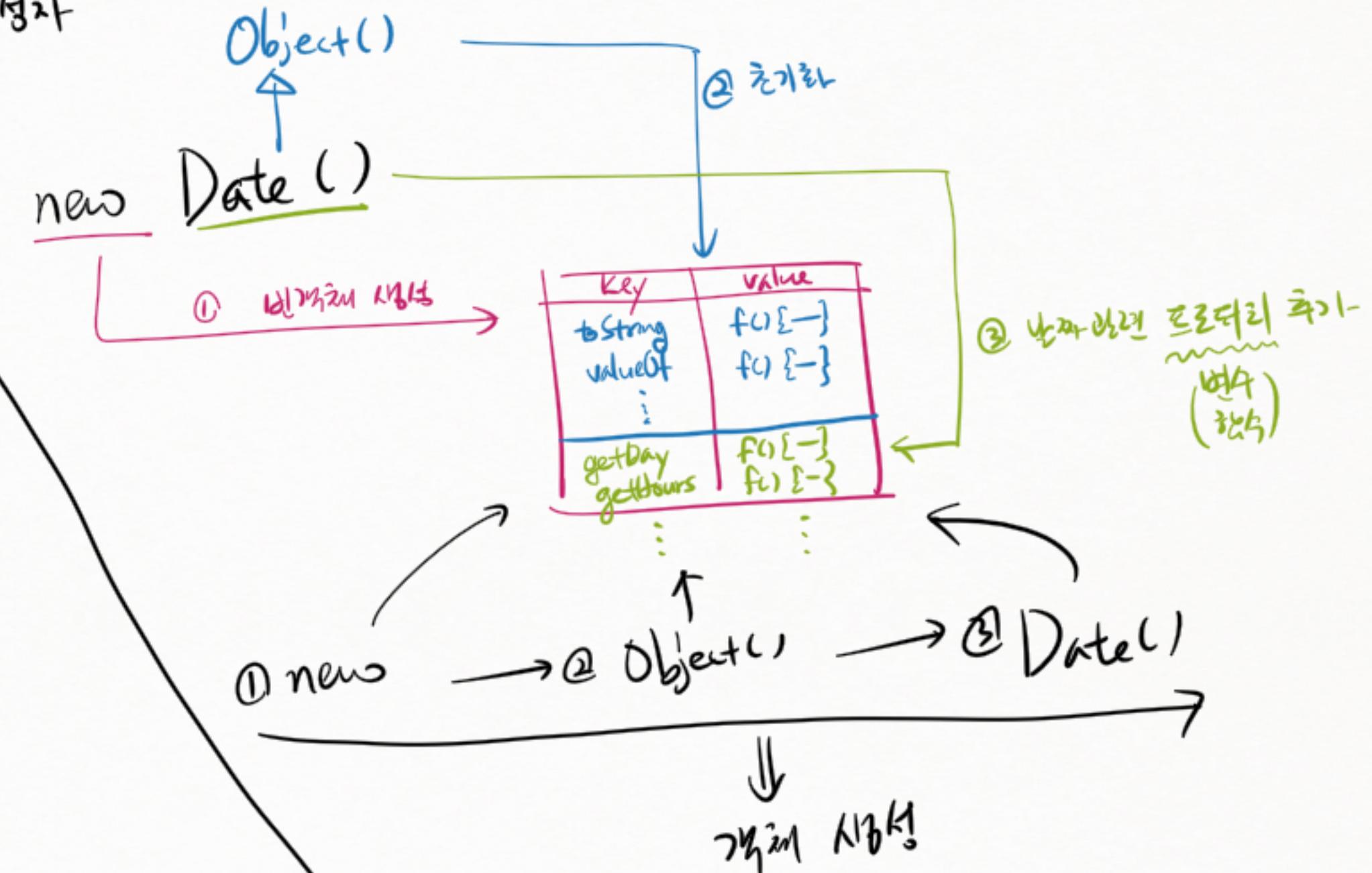
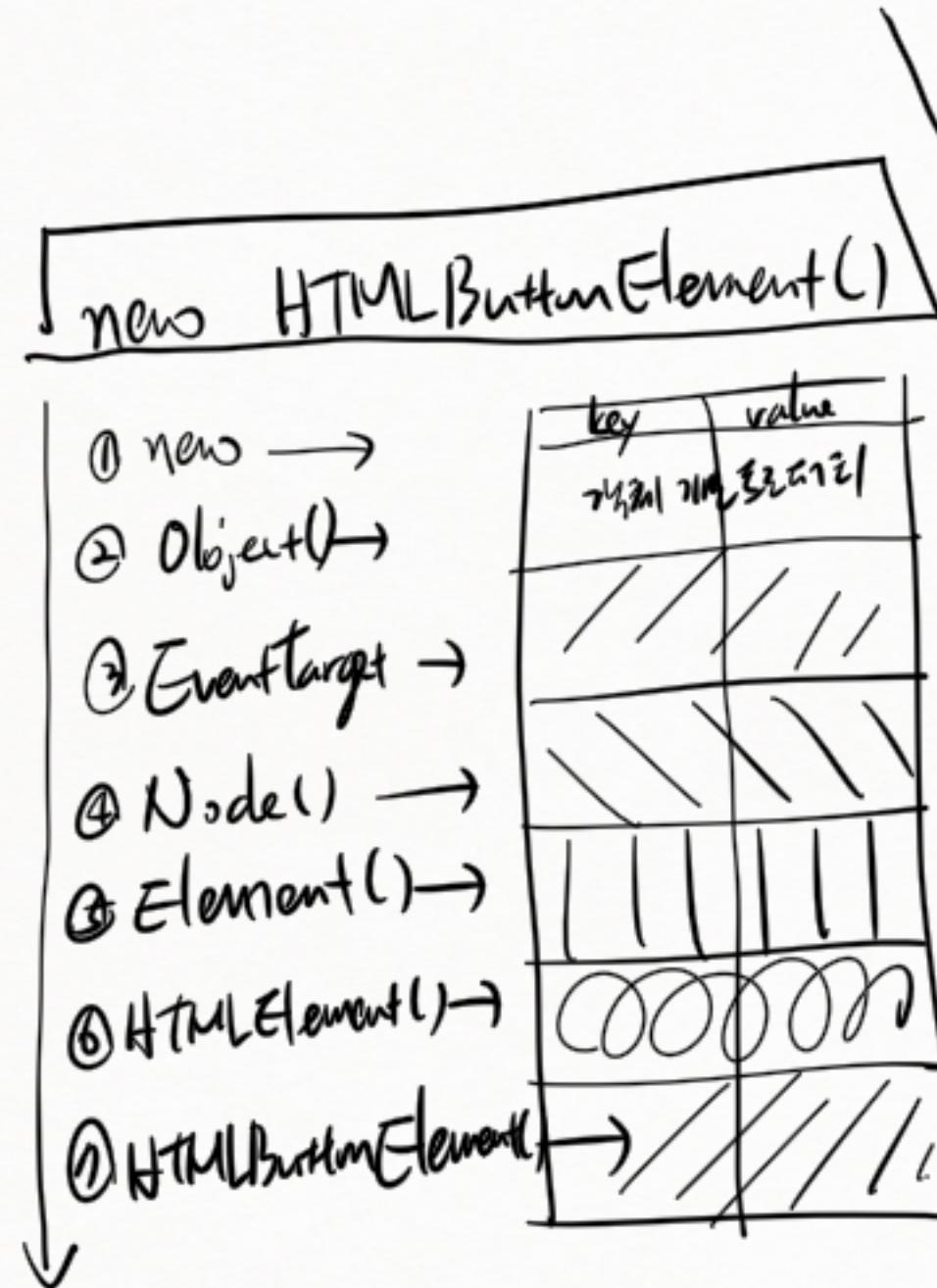
↳ basic, wbold, ...

↳ C, cobol, fortran

↳ C++, Java



* Date() 생성자



* 184/82

const arr = ["aaa", "bbb", true, 100];

```

    graph TD
        A[const arr = ["aaa", "bbb", true, 100];] --> B[new]
        B -- ① --> C[Array()]
        C -- ② --> D[{"length": 0, "push": ""}]
        D -- ③ --> E[{"length": 4, "push": ""}]
        E -- ④ --> F[{"length": 4, "push": "aaa", "0": "aaa", "1": "bbb", "2": true, "3": 100}]
    
```

The diagram illustrates the creation of an array in memory. It starts with the assignment statement `const arr = ["aaa", "bbb", true, 100];`. This leads to the creation of a new array object via `new Array()` (labeled ①). The constructor function `Array()` (labeled ②) initializes the array with a length of 0 and a push method. Subsequent steps involve setting the first four elements ("aaa", "bbb", true, 100) at indices 0, 1, 2, and 3 respectively (labeled ③). Finally, the array is shown in its completed state with a length of 4, containing the values "aaa", "bbb", true, and 100 at indices 0, 1, 2, and 3 respectively (labeled ④).

* for 반복문

※ property 와 배열

key	value
name	"김재호"
age	20
working	true
toString	function
:	

↑ 배열 형태로
("김재호", 20, true, function, ...)

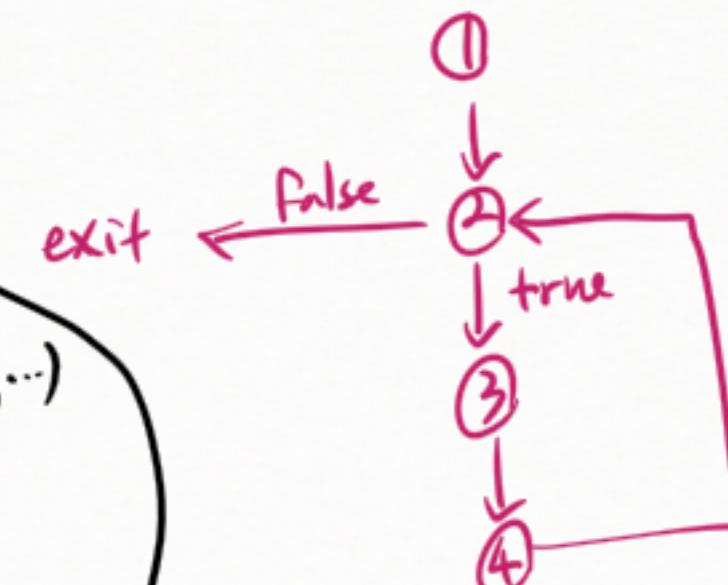
"Property"

for (변수선언 및 초기화 ; 조건 ; 변화값) {

 }

 ① ②

 ④



* 객체와 프로퍼티

const obj = new Object();

obj | 200

obj.name = "김길동";

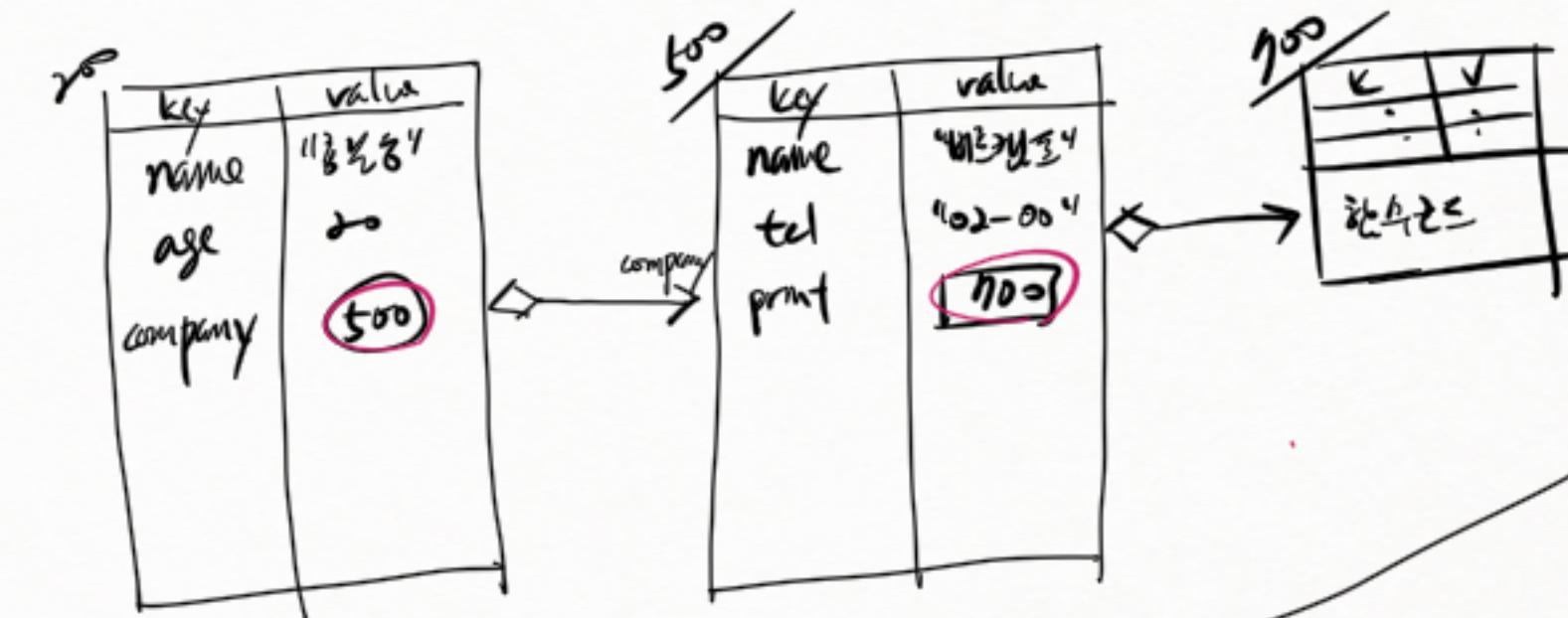
obj.age = 20;

obj.company = new Object();

obj.company.name = "비비드디자인";

obj.company.tel = "02-0000-0000";

obj.company.print = function(){};



Unified (연결된 언어)

Object Modeling (생각의 흐름을 그려내는 언어)

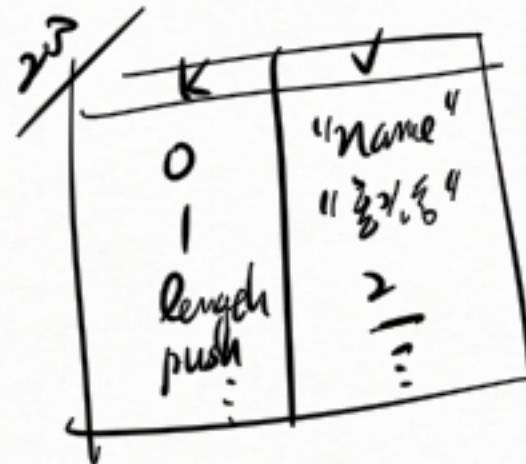
Language (언어, 문법)

작성하는 언어

* Unpacking destructuring

```
let a = ["name", "Bob"];
```

a 200



```
let key = a[0];
```

```
let value = a[1];
```

key

"name"

value

"Bob"

let [key, value] = ["name", "Bob"]; objekt



* destructuring: map

let arr = ~~["John", "111(-2222)", true, 20]~~;

arr
[200]

let [name, tel, working, age] = arr;

i l
let [name, tel] = arr;

* destructuring : 7'30"

```
let obj = new Object();
```

⋮
⋮

obj is null!

k	v
name	"John"
age	20
tel	"111-1111"
working	true

```
let {age, tel, name, gender} = obj;
```

if no value, undefined will be assigned
undefined

```
let {age, tel, ...other} =
```

k	v
name	"John"
age	20
tel	"111-1111"
working	true